Laboratory of Economics and Management
Sant'Anna School of Advanced Studies

Piazza dei Martiri della Libertà, 33 - I-56127 PISA (Italy)
Tel. +39-050-883-341 Fax +39-050-883-344
Email: lem@sssup.it Web Page: http://lem.sssup.it

LEM
Working Paper Series

Open Source Software: From Open Science to New Marketing Models

An Enquiry into the Economics and Management of Open Source Software

Paola Giuri[*]
Gaia Rocchetti[*]
Salvatore Torrisi [†]

[*] *Sant'Anna School of Advanced Studies, Pisa, Italy*
†University of Camerino and Sant'anna School of Advanced Studies

2002/23 **July 2002**

ISSN (online) 2284-0400

OPEN SOURCE SOFTWARE: FROM OPEN SCIENCE TO NEW MARKETING MODELS

AN ENQUIRY INTO THE ECONOMICS AND MANAGEMENT
OF OPEN SOURCE SOFTWARE

Paola Giuri, Scuola Superiore Sant'Anna, Pisa

Gaia Rocchetti, Scuola Superiore Sant'Anna, Pisa

Salvatore Torrisi, Università di Camerino and Scuola Superiore Sant'Anna, Pisa

Second draft: July 2002

Abstract

This research report analyses several issues on the economics and management of Open Source Software (OSS). It offers an historical description of the emergence of OSS and of heterogeneous types of actors involved in the OSS phenomenon. From a theoretical perspective the paper discusses some crucial topics: the organization and performance of the development process in open source and traditional proprietary models, the incentives to innovation in different regimes of intellectual property protection, the ingredients of business models based on open source software. From an empirical perspective it presents some data on the diffusion of open source software in some segments of the software market and develops a detailed comparison of open source and proprietary licensing models, also drawing some implications on the use of different types of open source licences in the commercial market.

Keywords: Open Source Software, Intellectual Property, Licensing, Business Model.
JEL Classification: O31, O34, L86.

Acknowledgements. We thank Giuseppe Attardi, University of Pisa, Alfonso Fuggetta, Politecnico of Milano and Guido Scorza, CIRFID, University of Bologna, for sharing with us their knowledge of technical and legal implications of open source software. We also benefited from information and discussion with Icube S.r.l., Ksolutions S.p.A., Microsoft Italia, and MLX S.r.l. (MadeinLinux) during the preparation of this report. The usual disclaimers apply.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Executive summary

This research report aims to analyse several crucial issues on the economics and management of open source software from a theoretical and an empirical perspective. The Report offers an historical description of the emergence of the open source software, and provides a comprehensive definition of the OSS phenomenon, which includes several characteristics (the development process, the licensing models, the different actors approaching OSS):

- OSS is a software whose source code is distributed with the object code;
- OSS can be distributed free of charge under terms of licences that guarantee the users the right to use, copy, modify and distribute the source code. But it is not necessarily *gratis*, because it is possible to sell the executable code and also to charge a fee for distributing the source code and for related services and support;
- the OSS development process is characterised by the possibility for anyone (developer, user) to download the source code, make modifications (further development of lines of code, debugging, etc.) and send them to the project leader without receiving any monetary compensation. The certification and distribution of certified modifications (add-ons etc.) is usually under the responsibility of a limited number of core developers and project leaders;
- several companies have entered the market since the late 1990s to assemble different OSS modules, to develop OSS interfaces that make the programs more user-friendly, to develop complementary products like drivers and other applications, to provide technical support and services (e.g., Caldera, RedHat, and Suse). On some occasions they have built their business models on popular OSS like Linux or Apache, while in other cases the business model was centred on new OSS invented by the company founders (e.g. Sendmail)
- Established proprietary software companies are changing their strategy both by revealing the source code of their products and by supporting OSS projects.

History

The historical evolution of Open source and proprietary software can be summarised with the following facts:

- During the 1960s and 1970s, it was quite common for software programmers working in academic institutions such as the MIT and Berkeley, or in corporate R&D laboratories like the Xerox Palo Research Center, to freely exchange their source code.
- At the same time, it was common practice for companies such as IBM to supply the code embedded in their hardware, without distributing it under specific license arrangements.

- Since the unbundling of software sales from hardware in 1969, a new generation of both proprietary, non freely available software was developed, with a significant amount of this software being developed by the users themselves.
- Until very recently, in-house development of software and embedded software (i.e., programs printed in various electronic devices) have represented a significant share of the total production of software.
- In the 1970s Unix operating system was invented at the AT&T's Bell Labs. Unix was freely available to the community of software developers. In the 1980s Sun Microsystems entered the market for Unix-based workstations and AT&T started to enforce its intellectual property rights over Unix.
- The FSF was founded in 1983 by Richard Stallman from the MIT to promote the GNU/General Public Licensing (GPL) scheme whose aims were to protect free software from being appropriated by commercial companies, guaranteeing the free distribution of source code, and the possibility to make copies, modifications and to distribute the software.
- In August 1991 Linus Torvalds was working on a free version of Unix kernel that could run on a PC and he posted an e-mail to a newsgroup asking for help in bug fixing. The Linux version number 0.01 was released in few weeks.
- In 1998 the Open Source Initiative (OSI) was created by a group of free software advocates. The OSI proposed the Open Source Definition and an OSI Certification Mark applicable to OSS licenses and introduced a new set of licenses that eases the commercial distribution of free source software.
- Today, new firms have entered the market to assemble different OSS modules, to develop OSS interfaces that make the programs more user-friendly, to develop complementary products like drivers and other applications, to provide technical support and services (ex. Caldera, RedHat, and Suse).
- Large incumbent proprietary software developers are revealing the source code for some of their products (e.g., Sun, Netscape-Mozilla and Nokia). Other large companies are supporting OSS by developing platforms running on Linux or other OSS (e.g., IBM, Apple).

Organisation of the process of development

Starting from the Raymond's work "The Cathedral and the Bazaar", which describes the development process of Linux compared to the traditional software engineering process, several people

within the OSS community have been claiming that OSS is more successful than traditional proprietary software in terms of efficiency of the process and quality of the product.

The reasons are found in the characteristics of the development process.

In the OSS model, open source developers are self-selected, OSS developers are free to express their creativity and are highly motivated by fun, while developers working in traditional groups perform their task mainly because of economic incentives; the intense activity of debugging makes software more reliable and its pace of evolution more rapid. The main strength of OSS is that it relies on a *potentially* large community of developers that constantly inspect code and fix bugs even though in fact the number of core contributors is limited.

The traditional model, is characterised by distinct phases that must be followed sequentially: a) analysis of users' requirements which describes the target customers' needs; b) a system level design that describes all modules composing a software and the way in which they interact within a common architecture; c) a detailed design that defines the characteristics of each module; d) codification or compilation of the code and integration of each module in a whole package (implementation task); e) testing that verifies if the software contains defects and bugs, followed by the process of correcting errors and re-design the whole system in order to accomplish with the change required by bugs fixing.

However it is important to note that the strength and weaknesses of the two models have not been tested yet. First, incremental, evolutionary change centred on sequential innovations is a key distinctive characteristic of software industry in general and it is still to prove whether open source software dramatically increases this feature.

Second, the evolution of software engineering, regardless of the openness of the source code, since the 1960s has produced variety of approaches and models of software development - from very structured ones, such as formal methodologies and integrated project support environments, to unstructured and flexible ones, such as rapid prototyping, incremental and evolutionary development, spiral lifecycle, rapid application development and extreme programming. The latter allow many forms of interactions among developers and a quite flat organisation of labour that stimulate creativity and are rewarding for individual developers.

Third, formal and structured methodologies for software development are adopted by a minority of large software developers (e.g., the so-called 'software factories' of hundreds or thousands programmers).

Finally, even if one admits than the bazaar style is more efficient thanks to its 'peer-review' feature, in theory it is possible to apply the same method to proprietary/commercial software either in house or by including users. Moreover, rising coordination costs may constraint the growth of OSS.

Therefore, further analysis is needed to provide substance to the claim that the open source software development process is more efficient than the traditional one.

Diffusion of open source software products

An analysis of several sources of data explore the diffusion of open source software compared to proprietary software.

The large number of projects hosted in the SourceForge web sites - over 40,000 - indicates the massive interest in open source amongst software developers. However, most of these projects are at a preliminary development stage. In fact, only 1.67 % of the projects are in a maturity stage and 15.02 % in a production stage. The largest share of the projects (28.16 %) are still in a planning stage. This static picture of the development stage of current projects does not say anything about their survival rate or about their probability of progressing in the stages of development. Moreover, even for the projects in the maturity stage, there is no evidence of the number of projects that are introduced in the market and that gained commercial success.

We focus the empirical analysis on the products that have reached the market and gained significant commercial success in selected segments of the software market. A synthesis of the main findings is the following:

In the *Web Server Software market*, Apache is the most adopted web server software on the public Internet since April 1996. Apache entered the market in 1995 and rapidly gained the leadership which was previously owned by the server software of the National Center for Supercomputing Applications (NCSA) of the University of Illinois. Microsoft entered later with the IIS product and became the second player in 1998. After an early entry and a rapid initial growth, Netscape has progressively loosed market shares while Zeus entered in 1997 and slowly gained the fourth position on active sites and very recently the third position on all sites.

Apache is the leader in Europe, Russia, Canada and India, while Microsoft gained the leadership in the United States and in China. Across Europe, only in Italy Apache and Microsoft shows very similar shares.

In the *Server Operating System market,* Windows NT/2000 has the largest share of shipments both in 1999 (38.4%) and in 2000 (40.9%). Linux, the second player with a share of 29.6%, grows faster than Windows in terms of shipments. Other server operating environments show declining shares and negative or constant shipment growth rates.

Data at the country level shows differentiated patterns. Linux is the most adopted operating system in the United Kingdom, France, Germany, Ireland, Portugal and Russia. Instead, in Italy and Spain Microsoft Windows is the leader. Solaris is almost always the third most used operating system, except in Germany, where Windows is the third OS with an 8% average share in the period and shows a decreasing pattern over time, and the United Kingdom, where Solaris and Windows compete head to head for the second position.

The time trend is also different across countries. In some cases (i.e., France) the growth of Linux has occurred along with a slowdown of Windows, while in others (e.g., Portugal) Linux has grown to the detriment of Solaris or other operating systems.

The analysis of the market shares of different platforms calculated on firms' *revenues* shows that in 1999 and 2000 the highest revenues have been obtained with Unix platforms, followed by Windows and Netware. Linux and other operating systems gained much lower revenues. However, Linux, Unix and Windows experienced positive growth rates of revenues from 1999 to 2000. Comparing these data with the shipments data, we may observe that although Linux experienced a high share of shipments, the revenues have been quite low.

Moreover, in 2001 Windows had the largest share of Integrated Collaborative Environments (ICE) revenues, followed by OS/400, Unix and Linux and other open source operating environments.

Finally, in 2001 66% of revenues from Web Server Software and Web Acceleration Software (WEBS) derived from sales of products running on Unix platform, 28.1 % on the Windows operating environment and lower shares in the other platforms. However, the market is expected to substantially grow from 2002 to 2006 and Windows, Linux and other open source platforms to grow faster than Unix, by hosting more WEBS and gaining largest market shares.

In the Italian market, the opportunities of revenues from Linux are growing and are expected to grow in the next years. The value added from the Linux software have grown at a growth rate of 44% between 1999 and 2000, 98 % from 2000 to 2001, and is expected to keep this pace in the subsequent years. The growth rate of services and software for dedicated environment have been much stronger and are expected to grow faster, suggesting that the opportunities for revenues in this market are larger for specific software development and services.

Intellectual property rights: a comparison between proprietary and open software

In the software industry the appropriability of innovation, including intellectual property rights (IPR), is quite weak. This is in part due to the relatively young age of this industry and in part to the characteristics of software technology.

Since the unbundling of software from hardware in 1969, which gave rise to the birth of an independent software industry, the issue of IPR in software has become the object of a lively debate between the advocates of a strong legal protection, which point out the importance of incentives to innovation, and the advocates of a weak protection, which highlight the social benefits of high entry rates, competition among different technological standards and diffusion of technological knowledge. In line with the supporters of a weak legal protection there is a stream of the literature which makes the point that in industries characterised by strong network externalities, such as operating systems, where *competition between alternative standards* is structurally weak, IPR should be weak to guarantee entry and competition among complementary technologies *within the standard* (e.g., application software) (Merges, 1996; Cohen and Lemley, 2001). The advocates of weak IPR claim that a strong IPR is not a necessary condition to guarantee a rapid rate of change in this industry since many important software innovations have been introduced before a strong IPR have emerged (see Merges, 1996, for a wider illustration of this point). On the other hand, they claim that a strong IPR may have negative effects on the rate of technical change since many important innovations in this industry have been introduced by new firms rather than established companies (Prusa and Schmitz, 1991).

Within the current legal system, copyright, patents and trade secrets are complement both in the US and in Europe in that the same software invention can be protected under these three laws. In particular, the absence (copyright) or weak disclosure obligations (patents) make it possible to cover under trade secrets the source code and to license the object, executable code against the payment of license fees.

The market for software is influenced by the property rights regime, which affects the scope of property rights (from patents to copyleft), and the contractual regime, which provides the institutional framework within which software technology can be transferred across individuals and organisations (e.g., by different licensing arrangements).

OSS is different from proprietary software on both grounds – the property right regime and the contractual regime. These differences are reflected in the organisation of marketing and distribution of software products and services.

Proprietary commercial software is distributed under licenses that usually limit the use, and deny the possibility to copy, modify and distribute (distribution is prohibited unless the distributor pays royalties to the copyrights holder). Furthermore, commercial proprietary software is usually protected by trade secret. Commercial software companies can then extract rents from their R&D activity, protecting their innovation from competitors. Although software is technically pure information, proprietary software can be produced and distributed as a partially private good thanks to IPR.

In licenses that accompany open source software, the copyrights holder maintains the right to use, modify and distribute the software but gives up the trade secret over the source code and allow users the same rights, therefore making the software a public good.

We describe four different types of software distribution (proprietary software, public domain software, Open Source/Free software, Shared source software) and compare them according to several characteristics of the adopted licence models.

Some general issues emerge from the comparison of these licenses.

First, while all OSS licenses overall guarantee the source code availability, proprietary software licences generally do not allow the access to the source code.

Second, proprietary licences do not provide the right to copy, modify and distribute the software, and impose several rules on the use. By contrast, OSS licenses ensure that users are free to use and copy the software without limitations.

Third, proprietary licenses seem to fit better with national laws than OSS in general. For example, the 'viral clause' of the GPL can conflict with the copyright law when the author of modifications or integration to the original covered code can demonstrate that those modifications are original intellectual works which rely on the original GPL covered code only to a limited extent.

Proprietary licenses written for specific categories of customers and different purposes may contain explicit rules for solving third party claims and litigations, conflicts and violations.

Finally it is important to remark that, even though OSS generally is not against commercial software, there are significant differences between proprietary and OSS licenses. Users typically do not buy a proprietary software, they only acquire the *right-to-use* against the payment of a royalty or a fee. Therefore proprietary software is 'owned' by the original developer who 'holds' the intellectual property protected by patents and/or copyright. The level of the right-to-use fees is a function of the perceived value of the software and the competition in the market for that particular software.

We also compare four OSS licences models (GPL, LGPL, MPL/NPL and BSD), also according to some properties specific to OSS licences.

The analysis suggests that OSS licences share most basic characteristics, except for a few cases, i.e. the BSD licence does not include initial developers special rights. Moreover, the conditions under which the distribution is allowed with the BSD are slightly different than with the GPL since the BSD does not include any obligation to distribute the source code nor limitations on the possibility to sell the software. In addition, the MPL includes explicit rules in order to solve third party claims and litigations, conflicts and violations. It is also more compatible with national laws both because it does not contain any 'viral' clause and because it explicitly claims that the license shall be subject to the jurisdiction in which the software is distributed.

The analysis of characteristics specific to OSS licenses suggests that licensing models like the MPL and the BSD seem to be more appropriate for business actors compared with the GPL licence, which does not allow to merge source code under GPL with proprietary software. As mentioned before, the 'viral clause' create problems for individuals and business firms that aim to develop or distribute software programs under different licence schemes. The LGPL license presents almost the same problems as the GPL but they can be used in commercial proprietary software that uses LGPL libraries without modifying or including them in the executable code.

By contrast, the MPL and the BSD licences do not impose any 'contamination' clause on OSS users and as such are compatible with proprietary software. Compared to the MPL, the BSD license does not impose restrictions on source code availability but, on the other hand, it has the lowest degree of protection of openness so that the code distributed can easily be appropriated by developers or distributors of proprietary software.

Finally, all OSS licenses models allow for distribution fees (not license fees) and therefore provide private incentives to commercial distributors.

The economics and business of open source software

Several recent works have studied the Open Source Software phenomenon focusing on different issues: the incentives to develop software without monetary compensation, the heterogeneity of users preferences that induce their involvement in the innovation process, the competition between OSS and proprietary software, the business models for making profits from open source software.

About incentives of OSS developers, the economic literature has tried to understand the motivations of actors by focusing primarily on the behaviour of hackers who participate in associations like the FSF and the OSI. The literature has highlighted the following main motivations: ethical and political reasons; reputation inside the community; need to adapt software available on the Internet to their own requirements and to solve specific problems; pure fun.

However, those motivations do not clarify completely some key questions: is open source software comparable to open science? If so, does it matter that a scientific discover in software be produced by a OSS institution or proprietary software one? The economic literature has highlighted some links of OSS with open science, mainly based on the fact that like scientists, OSS developers share a virtual context and a common language that favour the exchange of information and share incentives typical of the scientific community.

However, some models of competition between a community of philanthropic innovators (the OS community) and proprietary producers highlights two kinds of externalities arising from the OSS that is characterised by gift exchange and community values shared by philanthropic innovators. First, the

typical positive externalities which reduce the costs of proprietary software. In line with the new growth theory, these externalities ("manna from heaven") are positively associated with the number of goods (competitors doing R&D activities) in the market. Second, philanthropic innovators produce also negative externalities due to the low (around zero) prices of OSS. The low price reduces the demand for proprietary software and as a consequence profits of proprietary software shrink. Therefore, OSS 'steal business' from proprietary software. These negative externalities reduce the ex ante incentive to innovation of proprietary software firms and have negative consequences for growth in the long run (even if the 'manna from heaven' may have positive short term effects).

Another economic model of competition analyses the possibility of a new OSS standard to replace a dominant proprietary technology and the reaction strategies of incumbents.

Open source and proprietary companies compete through different strategies and business models.

In parallel with the diffusion of several new licence models associated with the OSS a variety of new business models have been introduced into the market. The fundamental ingredients of a business model in the software industry overall can be summarised as follows:

- The core business of the firm - new software development, improvement of existing products, software distribution, services;
- The organisation of the development process and the division of labour with other organisations– scale of the OSS developers team, type of software code disclosed and distributed, contractual agreements with suppliers and distributors;
- The sources of firm's revenues and related issues – software product pricing strategy, services billing strategy or and pricing of complementary products;
- The IPR strategy and the licensing model (Open source vs. proprietary).

The traditional business model is characterised by internal software development (or subcontracted to third parties). R&D investments represent a fixed cost that is recovered usually through proprietary licence fees. The licence provides the right to use the software, but not to copy, modify and redistribute it. The source code usually is not disclosed with the object code and a price is charged for the acquisition of a particular copy of the programme (license fee) or for the temporary use (renting) of the programme. A separate price is normally charged for support services. Prices in this sector are not driven by marginal costs. Price discrimination is often operated across different of users (for example business or academic customers) for the same product or service, according to the price elasticity of demand and the market power of customers relative to vendors. Finally, distribution and post-sales services are provided directly by the software company or by third parties – either affiliated vendors, exclusive agents or independent resellers. The value added by third parties varies with the type of

software distributed (its unit value, and the complexity of its installation and maintenance). In typical shrink-wrapped products, like PC operating systems and office automation applications, most of the value is produced by the editor (e.g., Microsoft, Symantec or Adobe).

As for OSS business models, there are various types of business models that differ mainly for the source of revenues (software development or services and distribution) and the licensing models:

1. business models centred on purely collective developed of software by a group of hackers (e.g., Linux or Apache). The source code is downloaded, assembled and sold together with support services by commercial distributors (i.e., RedHat, Caldera, Suse). The quality of the output is related to the individual and organisational capabilities of the group of developers (organizational features such as the number of project administrators and developers, the management style, and features related to the vitality of the project such as the number of fixed bugs, patches, external contributors etc.). The bulk of revenues come from services.
2. business models based on open source software initially developed by a company and disclosed to the community of OSS developers (e.g. Zope). The quality of the software depends both on the company initial R&D efforts and the community contributions. The revenues come mainly from software customisation and services.
3. mixed business models mostly adopted by commercial proprietary software developers. This category includes firms that make open the source code of products previously closed (e.g., Netscape-Mozilla). On many occasions, the firms that adopt this mixed approach develop both open and proprietary software. The revenues of this category of firms mostly derive from hardware sales or other activities such as book publishing.

The report provides a short description of some OSS business models adopted by Linux distributors, RedHat, Caldera, Suse, Turbolinux. A deeper understanding of these and other business models that have emerged over the last years requires an ad hoc analysis that is beyond the scope of this research.

2. The emergence of Open Source Software: history and the process of development

2.1 What is Open Source Software

Before the term open source software (OSS) became popular among business practitioners and scholars, in the software community the concept of free software was used to mean voluntary exchange of source code among developers working in different organisations (universities, public agencies and for profit institutions). During the 1980s free software has taken the form of an ideological opposition to proprietary software. As we discuss in subsequent sections, open source has recently emerged as a more pragmatic variant of free software. Despite the differences that we shall analyse later on, some scholars tend to consider open software as a synonym of free software. As a matter of fact, free software and open software are substantially two wings of the same movement which aims to promote a novel model of software engineering and new models of software distribution. Except for the following section, where we shall analyse the distinctions between free and open source software at length, in this work we adopt the term OSS to mean both free and open software.

As E.S. Raymond (2000) wrote "The ideology of the Internet Open Source Culture (…) is fairly a complex topic (…)". In fact, different experiences and different motivations are joined in this movement. To make this phenomenon even more complex, the contents of open source definition have changed during the last years. Moreover, the OSS world involves different actors with heterogeneous characteristics and motivations. Therefore, a simple definition of open source software would underestimate the complexity of the phenomenon, that we can summarise as follows:

- OSS is a software whose source code is distributed with the object code[1];
- OSS can be distributed free of charge under terms of licences that guarantee the users the right to use, copy, modify and distribute the source code. But it is not necessarily *gratis*, because it is possible to sell the executable code and also to charge a fee for distributing the source code and for related services and support;
- the OSS development process is characterised by the possibility for anyone (developer, user) to download the source code, make modifications (further development of lines of code, debugging, etc.) and send them to the project leader without receiving any monetary compensation. The certification and distribution of certified modifications (add-ons etc.) is usually under the responsibility of a limited number of core developers and project leaders;

[1] Source code is a computer program written by a programmer in a language readable by individuals; source code is input to a compiler or assembler, in order to derive object code, that is the machine readable code.

In what follows we outline a brief history of open source software by highlighting how the open source definition has evolved, and the motivations behind its changes (Section 2.2). Furthermore, we describe the technical and institutional characteristics of the development process and point out some puzzles that need to be solved through future careful empirical exploration (Section 2.3).

2.2 History of OSS: source code availability vs. trade secret practices

Over time, different actors have adopted heterogeneous behaviours with respect to the choice of disclosing the source code of software, depending on their identity, their incentives and also the historical and social context.

During the 1960s and 1970s, it was quite common for software programmers working in academic institutions such as the MIT and Berkeley, or in corporate R&D laboratories like the Xerox Palo Research Center, to freely exchange their source code. At the same time, it was common practice for companies such as IBM to supply the code embedded in their hardware, without distributing it under specific license arrangements. In the early days of general purpose computers for technical and commercial applications users gave important inputs to the development of software. For instance, the first operating system (called 'monitor') was developed at the General Motors Research Laboratories for the IBM 701 mainframe in 1955. Both users and computer manufacturers then actively participated to collective efforts aiming at developing and diffusing software. Through the 1960s computer manufacturers begun to systematically develop the operating systems and software development tools that helped users to develop their own applications (Torrisi, 1998, ch. 4). Obviously, users' and computer manufacturers' incentives to cooperate were different. Computer manufacturers' main objective was to promote the sales of hardware while users were interested in improving their applications running on expensive and unfriendly mainframes (Steinmueller, 1996).

The creation of ARPAnet in 1969 by the US Defence Department allowed continuous interactions to groups of developers, even though we do not know to what extent this and other defence-related projects implied free exchange of source code. A significant share of exchanged software for technical and scientific applications followed the invention of the Unix operating system by Ken Thompson at the Bell Labs in the early 1970s. Unix, entirely written in C language (invented by Dennis Richy in 1973), was further developed during the 1980s by different groups working separately for its portability on different types of machines[2]. The possibility to connect machines by the telephone line was an important innovation of Unix operating system. The creation of the USENET (more efficient than ARPAnet), facilitated the communications among software developers.

[2] For further details on the history of Unix development see McKusick (1999).

However, not every software circulating in those years was free and developed by the scientific community. Since the unbundling of software sales from hardware in 1969, a new generation of proprietary, non freely available software was developed, with a significant amount of this software being developed by the users themselves. Until very recently, in-house development of software and embedded software (i.e., programs printed in various electronic devices) have represented a significant share of the total production of software. As mentioned before, this large amount of non traded software has been facilitated by hardware manufacturers and by the increasing diffusion of high level languages and development tools.

In the 1980s two major events marked the story of free software. First, Sun Microsystems entered the market for Unix-based workstations and, second, AT&T started to enforce its intellectual property rights over Unix. Proprietary and non free software producers reacted to AT&T by founding different associations which supported various, mostly incompatible, versions of Unix. Among these, the Berkeley Software Distribution and the Free Software Foundation (FSF). Later on, in the 1990s, the Open Source Initiative was founded as an organisation distinct from the FSF.

The Free Software Foundation and Linux

The FSF was founded in 1983 by Richard Stallman from the MIT[3]. The FSF promoted the GNU/General Public Licensing (GPL) scheme allowing the protection of free software from being appropriated by commercial companies, guaranteeing the free distribution of source code, and the possibility to make copies, modifications and to distribute the software.

In addition, the GPL contains a 'viral' clause for which any work that is distributed or published, that in whole or in part contains or is derived from the GPL-ed code, has to be distributed as a whole under the terms of GPL, *de facto* preventing from every commercial utilisation of GPL-ed software.

The philosophy of the FSF and its main goal, as stated by Stallman in his GNU manifesto (see www.gnu.org), is the diffusion of software to guarantee the free access to knowledge to the maximum possible number of customers, and then to contrast monopolistic practices of some companies (like IBM in 1980s and Microsoft more recently) that may hamper people's freedom. Stallman emphasises the nature of software as scientific knowledge, not as a proprietary invention, and according to this view software has to be shared among users to improve social welfare. He named the software distributed under GPL '*free* software', underlining that *free* stands for freedom of use and copy while it does not mean 'free of charge'.

[3] For further details see R.M Stallman (1999) and http://www.gnu.org/ .

Since the 1980s Stallman and his community have actively tried to build a free version of the Unix operating system by massively re-writing free utilities which drew on Unix's utilities. By the early 1990s, however, Stallman and the FSF had not managed to write a free Unix kernel yet.

The advent of the Internet in the 1990s and the introduction of Linux impressed a dramatic acceleration to the diffusion of open software.

Linux was created by Linus Torvalds, a computer science student from the University of Helsinki. In August 1991 Linus Torvalds was working on a free version of Unix kernel that could run on a PC and he posted an e-mail to a newsgroup asking for help in bug fixing. He assured that he would have included in future versions new features developed by others as long as they would have also been freely distributable, according to the GPL licensing scheme.

The initiative had a great success, and the Linux version number 0.01 was released in few weeks (September 1991). Further releases followed at a rapid pace (http://www.li.org/linuxhistory.php) and the version 0.1 was released in December 1991.

In 1993 Torvald rewrote the kernel and a year later, when he released Linux 1.0, the operating system could compete in stability and reliability with the commercial version of Unix. Linux was rapidly adopted as Unix free kernel and attracted a large community of old and new hackers (Raymond, 1999a).

At the same time software vendors like Red Hat, Debian and Caldera created commercial distributions of Linux that bundled the operating system with utilities, applications and graphical user interfaces. In 1999 Red Hat went public successfully; and few months later LinuxCare, an important Linux service provider, announced several alliances with large IT companies such as IBM, Dell and Motorola.

The Open Source Initiative

In 1998 the Open Source Initiative (OSI) was created by a group of free software advocates. The OSI proposed the Open Source Definition and an OSI Certification Mark applicable to OSS licenses. The OSI introduced an important change to the GNU/GPL licensing scheme, introducing a new set of licenses that eases the commercial distribution of free source software. Eric Raymond, one of the most prominent founders of the OSI, and his collaborators aimed at making free software visible to a wider public and facilitating its diffusion in the business world (Raymond, 1999a). [4]

Raymond noted that Linux success contradicts the Brook's Law, according to which as the number of programmers involved in a project rises, the work performed scales up linearly, but complexity and vulnerability to bugs rises by quadratic terms. More generally, in his "The Cathedral and the Bazaar"

[4] See http://www.opensource.org/docs/history.html for a complete list of OSI founders.

(2001) Raymond describes the free/open source community and the reasons of its success. This book has helped to spread the free/open source philosophy within the hacker community.

In 1998 Bruce Perens wrote for the OSI the Open Source Definition (OSD) which was adapted from the Debian Software Distribution Guidelines (Perens, 1999). The OSD claims that OSS has to be distributed in source code, distribution terms may not restrict any parts from selling or giving away the software, or modified parts of it, as a component of an aggregate software distribution containing different software from several different sources. An OSI approved license cannot require a license fee and must not place restrictions on other software that is distributed along with the licensed code, for example the license cannot require that software distributed on the same medium must be open source[5] (see Box 1 for details).

Netscape Corporation was convinced by Raymond's statements that a Linux-like development process was the key for success and in 1998 decided to put in the web its Navigator to contrast Microsoft on the browser market[6]. Netscape experiment has not had the success that its supporters hoped, but the birth of the OSI and the focus they put on the possibility to commercialise open source software, avoiding the GPL 'viral' clause, attracted many other business companies in the open source world.

[5] Note that also the GPL complies with this requirement, by stating that only the code that forms a single work with GPL-ed one has to be distributed under GPL terms. See www.opensource.org/docs/definition.html for further details.
[6] For further details on the Netscape experience see Hamerly, Paquin and Walton (1999).

Box 1 . The Open Source Definition provided by the Open Source Initiative (OSI)

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

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

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavour

The license must not restrict anyone from making use of the program in a specific field of endeavour. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

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

8. License Must Not Be Specific to a Product

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

9. The License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software."

Source: http://www.opensource.org/docs/definition.html , Version 1.9

Today pure hackers, organised in foundations and non-for-profit organisations, are important members of the OSS development process. As Linux demonstrates, a relatively small number of core volunteers that take the lead in early stages of development can attract a critical mass of developers which in turn determine the take off of a project. As a matter of fact, two other categories of actors have joined the Open Source movement:

- several companies have entered the market since the late 1990s to assemble different OSS modules, to develop OSS interfaces that make the programs more user-friendly, to develop

complementary products like drivers and other applications, to provide technical support and services (e.g., Caldera, RedHat, and Suse). On some occasions they have built their business models on popular OSS like Linux or Apache, while in other cases the business model was centred on new OSS invented by the company founders (e.g. Sendmail[7])

- Large incumbent proprietary software developers that have begun to reveal the source code for some of their products (e.g., Sun, Netscape-Mozilla and Nokia). Other large companies are supporting OSS by developing hardware platforms (or complementary products) running on Linux or other OSS (e.g., IBM, Apple).

2.3 The Organisation of OSS Development

One of the first attempts to describe the organisation of an OSS development project has been done by E.S. Raymond in his seminal work "The Cathedral and the Bazaar". He describes the development process of Linux trying to understand the strengths of this new development style (the "bazaar") compared to the traditional software engineering process (the "cathedral"), and to identify the reasons of its success in terms of efficiency of the process and quality of the product.

A key characteristic of the Linux development process - as described by Raymond - is the role played by developers and users. The Linux community members (both users and developers) can download the source code, make modifications (further development of lines of code, debugging, etc.) and post it to the responsible of the project (i.e., Linus Torvalds). The project leader, along with core developers, examines the proposed modifications and releases a new version of the programme to the community. This organisation allows a rapid fixing of bugs, and the product evolves gradually thanks to the continuous feedbacks and improvements from different users and developers.

The active participation of users is not a novelty if compared to the Unix community's tradition for which the free exchange of code was a diffused practise. However, as argued by Raymond, the distinctive feature of the Linux process is that new releases including modifications, fixed bugs and improvements are distributed quite rapidly, and it is often the case that new releases contain bugs ("release early, release often"). This practice, which provides stimuli and rewards to the community members, is common to other OSS projects. Raymond compares the development process of Linux

[7] Sendmail ancestor called *Delivermail*, was originally written by Eric Allman while he was a student and staff member at the University of California at Berkeley. The first version of Delivermail was shipped with 4.0 and 4.1 BSD in 1979. Eric Allman stopped the development of Sendmail in 1982, and did not go back to it until 1990. In 1998, Sendmail Inc. was formed, to create a commercial version of the product.

(and OSS in general) to the traditional software engineering process, characterised by a rigid hierarchical organisation in which each group of tasks is associated to a team of developers.

He argues that the OSS development process is more efficient than the traditional one, and yields a higher level of product quality. He also argues that open source developers are self-selected, so that only the most talented people participate in the process. Furthermore, according to Raymond, OSS developers are free to express their creativity and are highly motivated by fun, while developers working in traditional groups perform their task mainly because of economic incentives. Finally, according to Raymond the intense activity of debugging makes software more reliable and its pace of evolution more rapid (Raymond, 1999b).

The open source software's advocates tend to apply Raymond's statements to the entire landscape of open source software, reinforcing the claim that the higher quality of the OSS development process depends primarily on the large amount of motivated people that inspect and improve the code. Moreover, the 'evolutionary' character of the process - i.e., the possibility to select among a great number of solutions proposed by highly skilled developers - guarantees that only the best solution among those proposed by developers will be integrated in each new release (Raymond, 2001).

Besides Raymond's quite 'ideological' statements, however, there is no reliable empirical evidence that demonstrates any significant association between open source code and quality of software programmes (Fuggetta, 2001). Raymond himself admits that he does not have any statistics that can show the alleged superiority of the open source software model, and can only induce this superiority from the evidence that OSS products evolve by improving their performance fast compared to proprietary software, that a large number of developers is involved in their development and mainly that they work on problems they care about and try to solve them "for love" (Raymond, 1999b). It is important to note that to compare OSS with proprietary software at this level of abstraction is very arguable from a methodological perspective. A rigorous comparison should be made between projects that are similar in all, or most respects (e.g. lines of code), but the fact that one is open and the other is proprietary.

The OSS development model is quite different from the traditional software engineering model (the so-called 'waterfall model'). The latter is characterised by distinct phases that must be followed sequentially. Vixie (1999) distinguishes among several different phases: a) analysis of users' requirements which describes the target customers' needs; b) a system level design that describes all modules composing a software and the way in which they interact within a common architecture; c) a detailed design that defines the characteristics of each module; d) codification or compilation of the code and integration of each module in a whole package (implementation task); e) testing that verifies if

the software contains defects and bugs, followed by the process of correcting errors and re-design the whole system in order to accomplish with the change required by bugs fixing.

According to OSS advocates, open source development represents "the best system-level testing in the industry" due to the fact that users not paying for the software are more friendly and tend to collaborate with the software distributor by signalling bugs, and are also more helpful if they can inspect the code and work on it (Vixie, 1999). Furthermore, the great number of users/developers guarantees that most flaws will be detected.

Only few OSS development processes contain all software engineering phases like, for example, the commercial versions of BSD, BIND, and Sendmail (Vixie, 1999). A typical OSS project generally is weak in both users' needs analysis and design stages, while its strengths lie in the implementation task. This is probably because the great majority of people involved in OSS development are voluntarily contributing. They do not have access to sophisticated software tools for software engineering, and often they are not interested in software design neither in users' needs analysis. According to some observers the design of an open source software is often implicit in the software itself and evolves over time along with the software. Therefore a design actually exists but it is not written down. Wilson (1999) argues that open source developers do not give the right weight to design first because in developers communities (both proprietary and open source) "heroic efforts by individual programmers had high status, while planning and testing did not" (Wilson, 1999). Moreover, many programmers do not have competence on software engineering either because they have not taken a software engineering course, or because courses do not actually teach them what they need for managing an open source software development process. However, this is going to change since large business companies have joined the open source movement (i.e. IBM, Netscape) and small companies like Red Hat entered into the market to commercialise open source product packages and to supply maintenance/support services (Wilson, 1999).

On the other hand, traditional development process is generally characterised by having both detailed marketing plan and system design (Vixie, 1999) although a lot of industrial software is still developed without following good software engineering methods (Wilson, 1999).

A typical description of the traditional development process - as in Raymond (2001) - is that the organisation is strictly hierarchical with each group of tasks associates to a team of developers. Vision and deliveries often depend on market requirements more than on users' needs. Also, programmers are less skilled than OSS developers, because a practice in the software companies is to bring people with low skills, give them an intensive course in one or more programming language without allowing them to learn how software has to be debugged (Raymond, 1999b). Moreover, they are less motivated than open source developers because their main incentive is money, and they cannot choose which part of

the software to develop or which task to perform. Therefore, they produce a software of modest quality (Raymond, 2001).

Again, these statements have not been tested yet. Moreover, the literature on OSS does not seem to account some important points. First, incremental, evolutionary change centred on sequential innovations is a key distinctive characteristic of software industry in general and it is still to prove whether open source software dramatically changes this feature. Moreover, as we discuss later on, other institutional innovations different from OSS could yield the same results as OSS (e.g., revisions of the patent system that allow reverse engineering rights and a weak application of the 'doctrine of equivalents'). Second, the evolution of software engineering, regardless of the openness of the source code, since the 1960s has produced a variety of approaches and models of software development - from very structured ones, such as formal methodologies and integrated project support environments, to unstructured and flexible ones, such as rapid prototyping, incremental and evolutionary development, spiral lifecycle, rapid application development and extreme programming. The latter allow many forms of interactions among developers and a quite flat organisation of labour that stimulate creativity and are rewarding for individual developers (Fuggetta, 2001). Third, formal and structured methodologies for software development are adopted by a minority of large software developers (e.g., the so-called 'software factories' of hundreds or thousands programmers). The majority of software firms, especially in the US and in Europe, are small, adopt 'ad hoc' and often chaotic development methodologies, and focus on loose management style, which give individual developers a great independence and creativity at the cost of inefficiency. Therefore, it is not true that proprietary traditional software as such implies a rigid and hierarchical division of labour which depress programmers creativity. Moreover, there is no evidence that a rigorous definition of development stages and traditional management practices are bad for quality. The Japanese 'software factories', for instance, are famous for their evolutionary, incremental approach to software development which involve large numbers of developers and a high level of product quality even though there are no examples of significant software innovations coming from those software factories (Bohem, 1981; Cusumano, 1991; Torrisi, 1998).

Finally, even if one admits than the bazaar style is more efficient thanks to its 'peer-review' feature, it is possible to apply the same method to proprietary/commercial software either in house or by including users. Moreover, there is not evidence that the only way to motivate people is distributing the source code (Fuggetta, 2001 p. 8).

Therefore, further analysis is needed to provide substance to the claim that the open source software development process is more efficient than the traditional one. The main strength of OSS is that it relies on a potentially large community of developers that constantly inspect code and fix bugs even

though in fact the number of core contributors is limited. For instance, in the case of Apache project (web server software) there are about 20 core developers around the world while in the Perl project (general-purpose programming language) there are 10-20 active programmers (including Larry Wall, the initial devoper) (Čubranić and Booth, 1999; Lerner and Tirole, 2000).

OSS advocates often argue that the characteristics of the OSS development process itself (i.e., source code availability, peer review, frequent release) guarantee that OSS products are more robust, reliable, and secure compared to proprietary software (Laing, 1999). Primarily, it is argued that the availability of source code and the possibility to inspect it provide the chance for defending against both accidentals fault due to bugs undetected, and against 'malicious code' hidden in the source code.

However, making a system's source code publicly available is not a sufficient condition to improve its security. First, source code availability may improve systems security only if the users have the capability and resources to find security bugs that make the system vulnerable and to fix them. Also, it could be quite hard for unskilled users to discover "back doors" in their system, even though today there are sophisticated tools that scan the source to identify potentially dangerous constructions (Viega et al., 2001). Second, it is possible to ensure source code inspection without giving up the intellectual property rights on the distributed software. Some traditional software producers have already changed their internal process for reviewing and testing the software by allowing qualified reviewers to inspect the source code under non-disclosure agreements[8]. Also, there are compilers that improve the security of the executable code without modifying the source code (Cowan, 1999).

By contrast, some argues that source code availability gives 'attackers' the opportunity to discover some exploitable flaws; in addition, attackers have the possibility to introduce 'malicious' code into the source code publicly available and freely downloadable from the Internet. As a matter of fact, OSS software official sites often strongly recommend to be careful in downloading unofficial releases[9]. On the other hand, it is argued that attackers can reconstruct any portion of executable proprietary code by reverse engineering and that even employees that work in big companies are able to introduce back doors in the software without software management's knowledge (Witten et al., 2001).

There is very scarce evidence on the quantitative measurement of the security of systems. Wheeler (2001) reports some data on the security of OSS and free software products compared to Windows (for example Microsoft IIS vs. Apache or Windows vs. Linux). For example, analysing Microsoft IIS security bulletins and Apache security advisories Wheeler found eight Microsoft bulletins concerning dangerous vulnerabilities for IIS, published from June 1998 through June 2001, against zero

[8] See for example Microsoft Shared Source Policy described in section 4.2.
[9] See for example Red Hat alert on http://www.redhat.com/support/alerts/security_gpg.html.

vulnerabilities for Apache in the same period. In addition, he reported that incidents, web sites 'defaced' (i.e. web sites whose content was illegally changed), virus attacks and, mainly, dangerous vulnerability are more frequent for Microsoft software than for the others. Therefore, he concluded that both OSS and Microsoft products have security vulnerabilities but OSS products seem to be less vulnerable.

However, the last Netcraft Survey published the 1st of July 2002[10] reports the existence of important vulnerabilities in both Microsoft IIS and Apache systems that could affect the 45% of Microsoft IIS sites and around 14 millions Apache sites (against 6 millions that have been upgraded and are now quite secure). Netcraft reports that "With over half of the Internet's web server potentially vulnerable, conditions are ripe for an epidemic of attacks against both Microsoft IIS and Apache based sites (…)".

Therefore, security seems to be a problem both for open source and proprietary web servers. According to Wheeler (2001), the Apache architecture guarantees that attackers have only limited 'privilege' on the system, so that they cannot erase or modify most files, while Microsoft IIS architecture allows the attacker the control of the entire systems (i.e. reading, modifying, and erasing any file on the system). However, one can argue that this weakness of Microsoft IIS depends primarily on its architectural design rather than the openness of its source code. At the same time, it is probably true that computers viruses attack more often Windows than Linux operating environments, but this may depend both on the fact that Windows is the most diffused operating system in the world. Moreover, some technical characteristics of Windows (for example the execution of macros in Word) may favour attackers regardless of the IPR under which Windows is distributed.

OSS advocates explicitly admit the OSS has also weaknesses. For example, the lack of an explicit design can imply that the project's quality is limited (Vixie, 1999). Moreover, the lack of an explicit design and of a vision for further development may limit the possibility that an open source product will experience a long term evolution.

Another possible weakness is due to the lack of strong economic incentives like those provided by traditional property right institutions (copyright and patents) which could limit the incentives to produce major product innovations. A preliminary, rough inspection of the largest OSS projects suggests that most open source products heavily draw on previous software developed by university or proprietary profit oriented institutions. The test of this hypothesis requires a more extensive empirical analysis that is beyond the scope of our work.

Čubranić and Booth (1999) observe that the OSS development process could even be faster and leaner than the traditional one, and could also yield more reliable products thanks to peer review. But

[10] http://www.netcraft.com/survey.

they also point out the rising coordination costs that constraint the growth of OSS. As a project gains success and the number of participants involved grows, the distributed division of labour typical of OSS tends to become inefficient. Therefore Čubranić and Booth suggest that more effective knowledge management and coordination techniques should be introduced in order to guarantee a sustainable growth of OSS projects. The organisational limits demonstrated by the OSS projects are not new to traditional software engineering. However, the literature generally points out that when a software project grows in size, it becomes difficult to manage and, after a critical threshold, the lines of code cannot grow more unless the overall design is reorganised (Lehman, 1985).

In contrast with this hypothesis, Raymond (2000, 2001) claims that in fact there are not strong limitations to OSS development. He argues that heterogeneous developers interact through the Internet according to consolidated rules that are defined by the community as a whole or by a leader - i.e. rules that signal errors and possible developments, rules to submit to the community solutions to problems posted by other developers, corrections and modifications, rules of acceptance of such modifications and incorporation into the 'official' releases of the product. In general the leader, or the group of the leaders, has a *vision* of the process development and gives the directions for further development by establishing, for example, which contribution may be incorporated in the software and which feature has to be developed. Thus, there are common rules followed by all participants. Each contributor recognises the role of the leader, or of the board of direction, and follows his indications. The point made by Raymond, however, confirms that the size of virtual development teams, like that of traditional teams, is limited by coordination costs and other factors.

These limitations keep the number of relevant contributors relatively small and impose a well defined division of labour among developers. Most successful projects - i.e. Linux (Godfrey and Tu, 2000), Apache (Mokus *et al.* 2000), and GNOME (Koch and Shneider, 2000) - are centred around a small group of core developers that shape the direction, coordinate the development efforts, decide which part of code has to be developed, evaluate and approve modifications submitted by other members of the community, while the decision to perform each particular task is voluntary.

In the case of Apache only the members of the board have the possibility to modify the official release and to distribute it. Another case in point, illustrated by Koch and Schneider (2000), is GNOME which has grown dramatically since its start in January 1997 to about 1,800,000 lines of code in November 1999. Koch and Schneider construct their metrics by calculating the difference between the lines of code added to GNOME files and the lines of code deleted during the period under analysis, taking data from GNOME CVS repository (i.e. the version control system of the project). It is important to remind, however, that these data include also commentary lines of code (Koch and Schneider, 2000, p. 2). According to Koch and Schneider, GNOME project has generated many

separate and self-contained sub-projects - each started at a different date and characterised by a diverse rate of growth. Some modules, like the core module, seem to have stabilised their size while others (i.e. gnome-libs) have grown. Thus it is possible that the total growth of GNOME project has been led by the growth of specific projects in different times (Koch and Schneider, 2000, p. 9-10).

Linux shows a similar growth pattern as GNOME. Godfrey and Tu (2000) analyses the Linux development process and show that Linux full distribution size has experienced a strong growth rate despite Linux is a large software system developed by a process lacking of traditional management tools. They measure the Linux size both by the number of byte of compressed file for the full kernel release (including all source of the kernel, documentation on kernel, script, and other utilities – but excluding executable files) and by the number of lines of codes extracted from the compressed file.

However, they also show that Linux's growth has been mostly accounted for by addition of new features and support for new architecture. For example, by subdividing the Linux kernel in subsystems they observe that the drivers subsystem (i.e. the collection of software with the function of translating each request the operating system does into a task that the hardware can execute efficiently) is the largest one and grows very fast (from less then 100,000 lines of code in June 1994 to about 1,000,000 lines of code in December 1999) while kernel growth rate was more limited (from less then 100,000 lines of code in June 1994 to less then 200,000 lines of code in December 1999). They observe that the growth and size of the driver subsystem distorts the idea of how large and complicated the Linux system is (Godfrey and Tu, 2000, p. 139). In fact, several device drivers are usually supplied in each kernel distribution (and most of them are large and complicated piece of software), one or more for each type of hardware configuration. Then, according to Godfrey and Tu, Linux is not so large and complicated as it seems since more then fifty per cent of it consists of device drivers that are external to the kernel. Also, the drivers subsystem experience a "super-linear" growth rate while Linux kernel its not growing so fast.

In many cases, like Zope (Fuggetta, 2001), Mozilla, Jikes Java compiler (Godfrey and Tu, 2000), and Jun 3D Multimedia Library (Aoki et al., 2001), software development is mostly performed in house and the source code is released for further improvement or bug fixing.

The limited size of OSS projects is confirmed by Krishnamurthy (2002), who has recently conducted a survey of the 100 most active projects classified by Sourgeforge as mature stage software (about two years of life). On average the projects analysed were founded in October 2000, and for most of them have been released several versions. These projects overall account for about 20% of all mature programs hosted in the Sourceforge website (www.Sourceforge.net). This survey shows that the great majority of mature projects are developed by a small number of programmers: the mean number of developers is 6.61; 29% of these projects had more than five developers; 22% of projects had only one

developer; and 51% of the projects had one project administrator. Furthermore, Krishnamurthy counts the number of mailing list and discussion groups associated to each product, plus the number of messages for each products (information that are available from the Sourceforge web site) and observes that the majority of these project did not produce massive discussions and exchange of messages through forums and mailing lists (33% of these projects had 0 messages). As the author argues, it seems that most OSS programs are developed by individuals rather than communities. But apparently much of these information flows do not occur through the Internet. We ask then how Kryshnamurthy knew about these flows and what are the channels through which these information flow.

To conclude, the experience of successfully products – like GNOME, Linux and Apache - cannot be taken as an example of the fact that open source development guarantees always greater efficiency and quality. For instance, Linux's success is due to project-specific characteristics, such as a large pool of talents that contributed to develop it, a high modularity of its design (which arises from Unix) and the great capabilities of its leader. These characteristics are not easy to replicate in all circumstances. Linus Torvalds developed the basic design of Linux by reusing many concepts from Unix. At the time of development these concepts were common knowledge among software developers and Torvald took advantage of the intrinsic modularity of Unix to modularise its project by defining the functions to be developed and the guidelines for each module. The developers who joined this project knew exactly what to do and how to do it because the architecture of the system was well defined and well known (Fuggetta, 2001, p. 9). Therefore the success of Linux is not enough to conclude that open source software process is better, neither it demonstrates that it is enough to post on the web a piece of software from 'scratch' to receive many useful responses that lead to a complex product such as Linux.

Moreover, the open source development process is characterised by a well defined organisation which is different from the typical bureaucracy and is more similar to a meritocracy where the best developers coordinate the process and leadership is legitimated by technical and managerial abilities (Mockus *et al.*, 2000).

3. Diffusion of open source software products: an international overview

3.1 Introduction

This section presents some key empirical facts about the diffusion of open source software compared to proprietary software. It describes some patterns of software adoption in specific market segments (operating systems, web servers, and specific applications) and provides a few data on the emergence of new commercial software distributors. Moreover, it exhibits some data on the revenues gained within different operating platforms by software companies and commercial distributors.

Our overall analysis focuses on the world market while some disaggregated data concerning operating systems and web servers segments are available for the largest European countries. Finally, more detailed analysis on the patterns of adoption is also conducted for Italy.

It is important to point out that there are some inherent difficulties in collecting and comparing data on software adoption, market shares, and revenues of proprietary and open source software. This is due to several reasons. First, most open source software can be freely downloaded from the web or is freely distributed by computer science magazines, and can be legally installed on many machines. Moreover, software that is sold through traditional distribution channels (in the form of packages contained in electronic media such as CD or DVD), and/or is downloadable from the web for a fixed amount of money can be privately copied or installed on many machines (illegally for proprietary commercial software, legally for open source software)[11]. This makes it difficult to estimate the number of adopters, especially in the low end market of PC operating systems and applications.

Second, the price of open software can vary across different distributions. It can be zero or include only the pure cost of distribution (CDRoms, boxes and instruction materials). Or, it can be virtually impossible to determine if the software is bundled with services (e.g., installation, integration and training) or sold in a single package with other costly software products. This makes it difficult the estimation of market shares and revenues from software and services, and the comparisons of market shares and revenues of companies adopting different business models.

In this context, most of the available data come from 'ad hoc' surveys on software users, producers or distributors, or from web-based surveys that collect information on web servers or other kind of

[11] As an example consider a Linux distribution such as Suse Linux. The package sold by Suse includes a CD (or DVD) containing both open source software in object and source code (for example the operating system Linux, libraries, office automation applications, utilities, games, drivers,…), and proprietary software. Primarily, it contains a proprietary "installer" - distributed only in object code - that links together all Linux components facilitating the act of installing Linux on a computer. Furthermore it contains freeware and shareware software (see section 4.1 for the software type definitions). The software included in the CD - both open source and proprietary - can be freely installed on several machines. For further details on product andservices sold by open source business companies see section 5.5 on business models.

software from different web sites. The data presented in this section have been drawn from various sources which adopt different methodologies. This heterogeneity adds further complexity to our analysis.

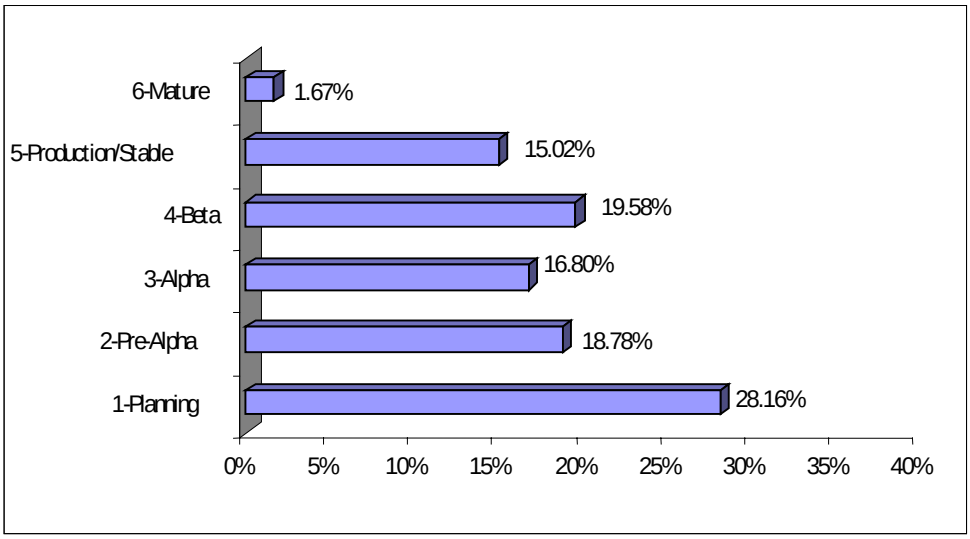
3.2 Open source software in the community of software developers

To analyse the importance of open source we start by investigating the number of open source projects in the world community of software developers. Some of these projects have yielded products that have not reached the market yet, others have reached the market only recently, while others are largely diffused. The market diffusion will be illustrated in the next section. This section focuses on a preliminary investigation of open source development and draws on information collected from SourceForge (www.sourceforge.net), the largest worldwide repository of open source projects. SourceForge collects source code and applications available on the Internet and provides free services to open source developers, including project hosting, version control, bug and issue tracking, project management, backups and archives, and communication and collaboration resources.

In this section we present some general statistics that illustrate the gross diffusion of open source projects, and the distribution of these projects according to some dimensions like the stage of development, the content, and the type of license adopted.

The large number of projects hosted in the SourceForge - over 40,000 - indicates the massive interest in open source amongst software developers. However, most of these projects are at a preliminary development stage. As Figure 3.1 clearly shows, only 1.67 % of the projects are in a maturity stage and 15.02 % in a production stage. The largest share of the projects (28.16 %) are still in a planning stage. This static picture of the development stage of current projects does not say anything about their survival rate or about their probability of progressing in the stages of development. A deeper analysis of their life cycle could provide more detailed data about the patterns of mortality of these projects.

Figure 3.1. Development status of Open source projects



Source: Elaboration from www.Sourceforge.net.

Table 3.1 shows that the most frequent objects of open source projects are Internet software, System software, Software Development technology, and Games/Entertainment software[12].

Table 3.1. Distribution of open source projects by topic

Topic	n°	%
Internet	7692	17.49%
System	6288	14.30%
Software Development	5303	12.06%
Communications	5028	11.43%
Games/Entertainment	4804	10.92%
Multimedia	3800	8.64%
Scientific/Engineering	2489	5.66%
Database	1943	4.42%
Office/Business	1452	3.30%
Desktop Environment	1213	2.76%
Education	909	2.07%
Security	861	1.96%
Other/Nonlisted Topic	840	1.91%
Text Editors	840	1.91%
Terminals	204	0.46%
Printing	139	0.32%
Sociology	90	0.20%
Religion	85	0.19%
	43980	**100.00%**

Source: Elaboration from www.Sourceforge.net.

[12] Note that totals for each table are different because there can be multiple entries, that is a project can be classified in more than one topic, while it is less likely that it refers to more than one licence scheme.

Table 3.2. Distribution of open source projects by type of licence

LICENCE	N°	%
Public domain	857	3.08%
Other/Proprietary licence	551	1.98%
OSI approved licenses	26444	94.94%
TOTAL	**27852**	**100.00 %**

Source: Elaboration from www.Sourceforge.net.

Multiple-dimensional analysis like, for example, that yielded by simple cross tabulation of objective and stage of development, can provide interesting insights into the patterns of open source software development. Moreover, more specific analysis on the starting time of each project can allow more detailed analysis of the life cycle of projects and their probability of reaching a mature stage. Furthermore, SourceForge provides data on the size of the projects in terms of people involved (both administrators and developers), operational characteristics of the projects like the vitality, as dependent on several dimensions like the number of times projects are viewed and downloaded, the bugs fixing activity, the number of patches requested and delivered, the requests for new features, the messaging activity in the mailing lists and so on, that we could use in our future research on the quality and performances of OSS. Finally, a comparison of project-level data with market data on the diffusion of open source products could provide further evidence about the commercial success of different open source projects and could also help to understand some success factors that are specific to the development process, such as the number of contributors and the organisation of development activities.

3.3 Open Source Software Products: Web Server Software and Web Platforms

This section introduces the analysis of open source products that have reached the market and gained significant commercial success. They represent then only a subset of the outputs yielded by the projects analysed in the earlier section.

We start our analysis with web server software and operating systems which include various categories of *system infrastructure software* (EITO, 2001). More precisely, web server software includes software that is run by web server computers connected to the world wide web. A web server receives client requests from an Internet browser, locates the resources and make possible the access to the requested resources by carrying out tasks such as security and authentication of users, shopping carts provision for e-commerce and access to databases. Operating systems are a fundamental component of

system-level software that serve the function of operating the hardware platforms and communications networks (e.g., LANs)[13].

Web server software

An important source of information on installed web server software is represented by Netcraft Web Server Survey (Netcraft, 2002a). Netcraft has been running its web server survey since August 1995. The survey reports by country and by region monthly analysis of web server software and operating systems of web sites and Internet connected computers worldwide. The methodology used in the survey is described at the Netcraft web page http://www.netcraft.com/survey/index-200007.html#active:

> "The Web Server Survey has run since August 1995, exploring the internet to find new web sites. At the end of each month, an HTTP request is sent to each site, determining the web server used to support the site, and, through careful inspection of the TCP/IP characteristics of the response, the operating system.
>
> Over the last two years there has been significant growth in the internet's DNS, fueled by rife domain name speculation, falling registration prices, easier and more efficient administrative procedures, and widespread publicity. It has become more common practice for companies offering internet registration services to place a template site on the web for each domain that they register. Additionally, some hosting service companies find it convenient to create new sites at the time of customer signup, rather than at the point at which the customer is prepared to place real, personalised content on to the web.
>
> So, whereas in the early days of the web, hostnames were a good indication of actively managed content providing information and services to the internet community, the situation now is considerably more blurred, with the web including a great deal of activity, but also some considerable quantity of sites untouched by
>
> human hand, produced automatically at the point of customer acquisition by domain registration or hosting service companies.
>
> The biggest domain registries are large enough to be significant even in the context of the 17 million sites found by the June 2000 Web Server Survey. For example, register.com host some 1.4m domain names, the great majority of which are template sites. Network Solutions have a system hosted at Digex which hosts around 750,000 template sites for domain name holders. These two domain name registries presently account for about 12% of the hostnames found in the Web Server Survey.
>
> Additionally, many companies will register in more than one domain. For example, Netcraft holds the netcraft.com, netcraft.net, and netcraft.co.uk domains, and currently uses three hostnames that will resolve to the Netcraft site. This means that there are nine names in the DNS that will resolve to the same content;
>
> www.netcraft.com, netcraft.com, ssl.netcraft.com, www.netcraft.net, netcraft.net, ssl.netcraft.net, www.netcraft.co.uk, netcraft.co.uk, ssl.netcraft.co.uk,
>
> Circa 1996-7, the number of distinct IP addresses would have been a good approximation to the number of real sites, since hosting companies would typically allocate an IP address to each site with distinct content, and multiple domain names could point to the IP address being used to serve the site content.
>
> However, with the adoption of HTTP/1.1 virtual hosting, and the availability of load balancing technology it is possible to reliably host a great many active sites on a single [or small number of] ip addresses. For example FreeServe has

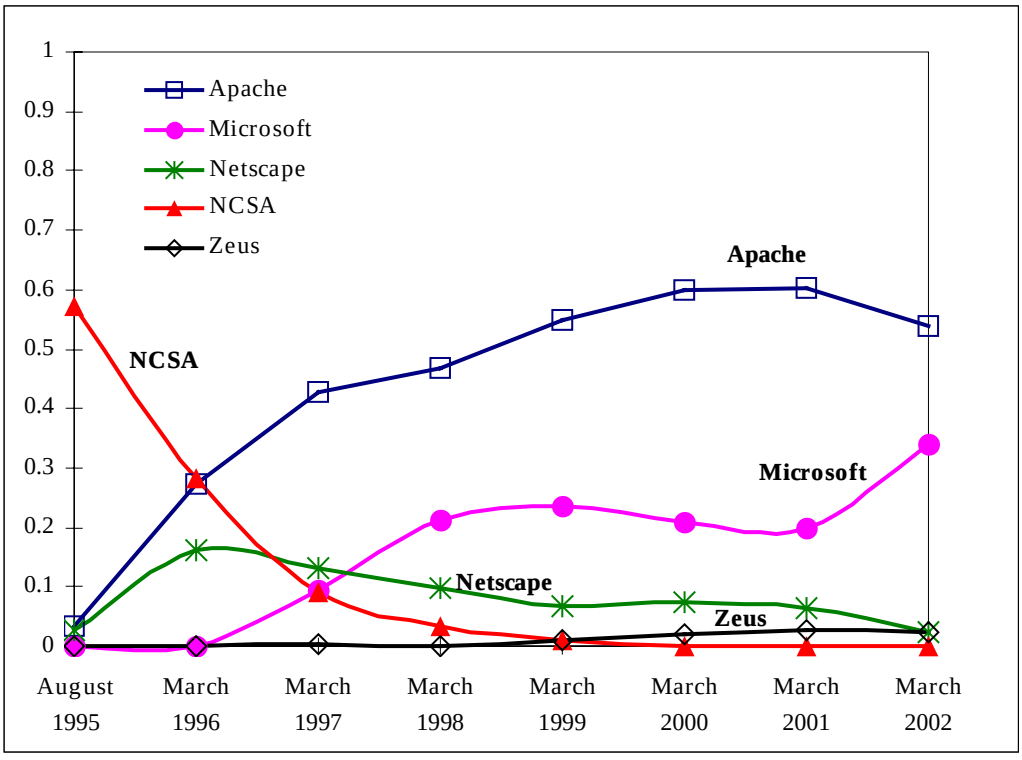
[13] A web server is "a server on the internet that holds World Wide Web documents and make them available for viewing by remote browsers" (http://www.computeruser.com/resources/dictionary). Business companies and organisations may either host their web sites on their own web server machines or ask to a hosting company that supplies the physical machine and the management of the website. Typically private citizens have their web site hosted by an Internet Server Provider (ISP). On each web server machine is installed a server operating system (i.e. Microsoft NT, Linux, Unix, Solaris); a web server software (i.e. Apache, Microsoft-IIS, Netscape) run on the operating system regulating the information flows directed to and from the Internet.

around 150,000 sites hosted on four load balanced IP addresses. These are substantially all active sites produced by real people crafting HTML in FrontPage, Word, Netscape, text editors, etc"[14].

The latest Netcraft Web Server Survey published in April 2002 shows that Apache is the most adopted web server software on the public Internet since April 1996 (Figure 3.2). Apache entered the market in 1995 and rapidly gained the leadership which was previously owned by the server software of the National Center for Supercomputing Applications (NCSA) of the University of Illinois. Microsoft entered later with the IIS product and became the second player in 1998[15]. After an early entry and a rapid initial growth, Netscape[16] has progressively loosed market shares while Zeus entered in 1997 and slowly gained the fourth position on active sites and very recently the third position on all sites. As it is described in detail in Table 3.3, in April 2002 Apache had a market share of 56.38% on all sites and 64.38 % on the active sites, followed by Microsoft with 31.96 % on all sites and 27.15 % on active sites.

Over the last year, Apache has lost its share of total sites, while Microsoft has increased its share. In the Active Site market, however, the pattern is the opposite.

Figure 3.2 Web Server Software Market Shares Across All Domains (August 1995-March 2002)



Source: Netcraft (2002a)

[14] Further technical details on the detection of active sites are available at http://www.netcraft.com/survey/index-200007.html#active and reported in Netcraft (2002c).
[15] Microsoft is the sum of sites running Microsoft-Internet-Information-Server, Microsoft-IIS, Microsoft-IIS-W, Microsoft-PWS-95, & Microsoft-PWS.

Table 3.3 April 2002 Web Server Software Market Shares

	All sites	%	Active Sites	%
Apache	21191595	56.38	10509138	64.38
Microsoft	12014054	31.96	4431875	27.15
Zeus	850956	2.26	182918	1.12
Netscape	832474	2.21	278775	1.71
Others	2696154	7.17	920901	5.64
Total	**37585233**	**100**	**16323607**	**100**

Source: Netcraft (2002a)

Another survey of Web server software from May 1998 to date is provided by Security Space (E-soft). The methodology of Security Space differs from that of Netcraft because Security Space polls the 'well known' web sites. The latter are those to whom at least another web site, which is also well known (which in turn is a site to whom other sites are linked and so on) is linked. Because of this methodology they actually visit only about 10% of all web sites. By focusing on a limited number of interconnected sites this survey is most probably biased in favour of a limited number of technologies which are adopted by organisations which share similar technological strategies.

Despite these potential limitations, the Security Space Web Server Survey published on April 2002, confirms the leadership of Apache across all domains (Figure 3.3). Moreover, unlike Netcraft data, these data show a steady increase of Apache's market share and a decline of Microsoft between 2001 and 2002. They also show some differences across domains and particularly across country domains. Apache is the leader in Europe, Russia, Canada and India, while Microsoft gained the leadership in the United States and in China. Across Europe, only in Italy Apache and Microsoft shows very similar shares. In Italy in May 2002 Apache has a market share of about 50% while Microsoft has 44.5 % (Figure 3.4, Figure 3.5, Figure 3.6).

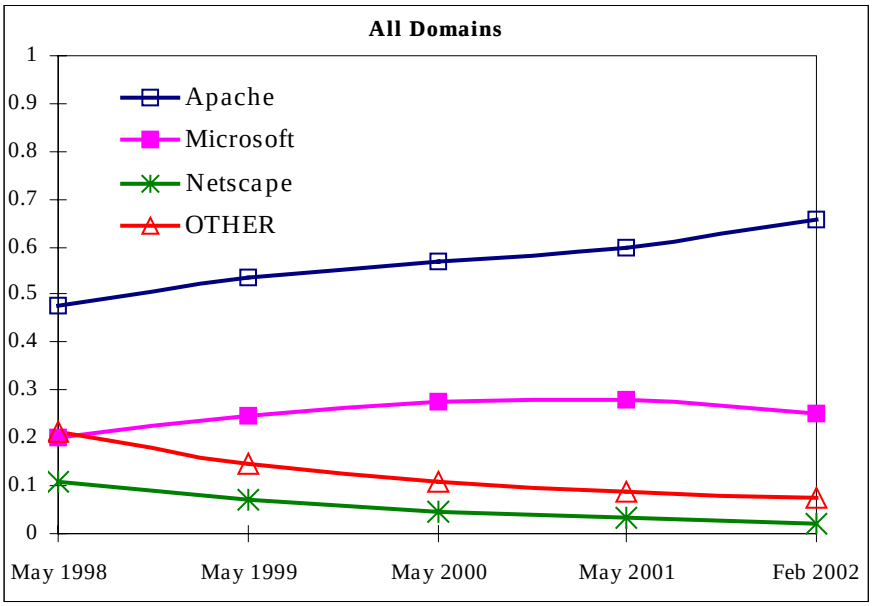
[16] Netscape is the sum of sites running iPlanet-Enterprise, Netscape-Enterprise, Netscape-FastTrack, Netscape-Commerce, Netscape-Communications, Netsite-Commerce & Netsite-Communications.

Table 3.4 Web Server software Market shares, March 2002

	All Domains	%	Italy	%
Apache	3899382	65.64%	34205	49.80%
Microsoft	1503177	25.30%	30536	44.46%
Netscape	110758	1.86%	767	1.12%
Zeus	62570	1.05%	104	0.15%
WebSTAR	54583	0.92%	479	0.70%
WebSite	28588	0.48%	305	0.44%
thers	281207	4.75%	2285	3.33%
Total	**5940265**	**100.00%**	**68681**	**100.00%**

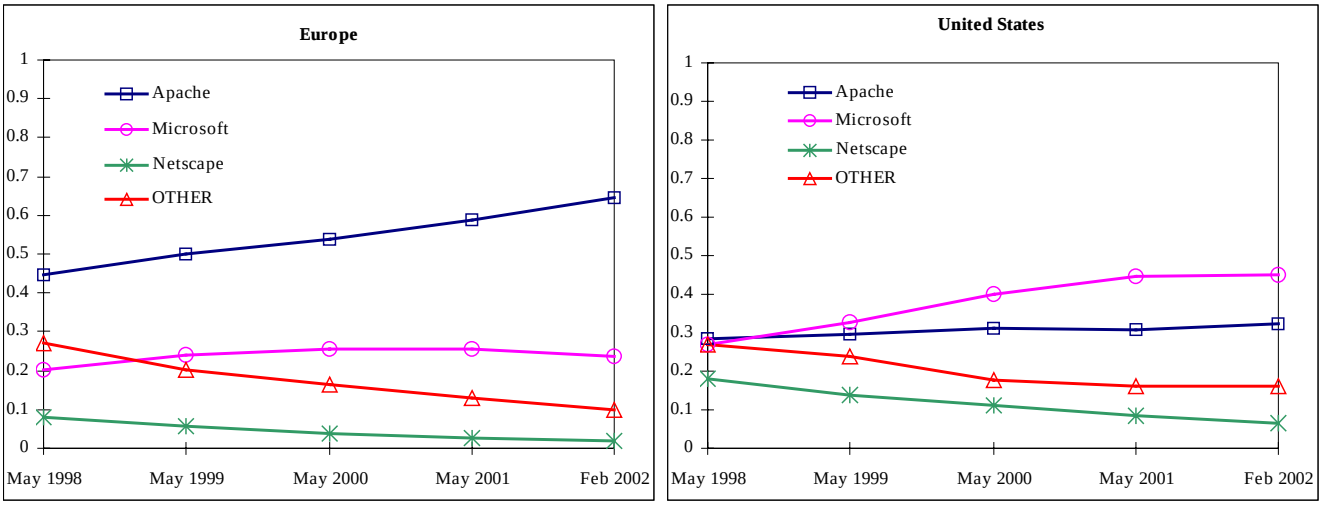
Source: Security Space (2002)

Figure 3.3 Web Server Market Shares Across all Domains (May 1998 - February 2002)



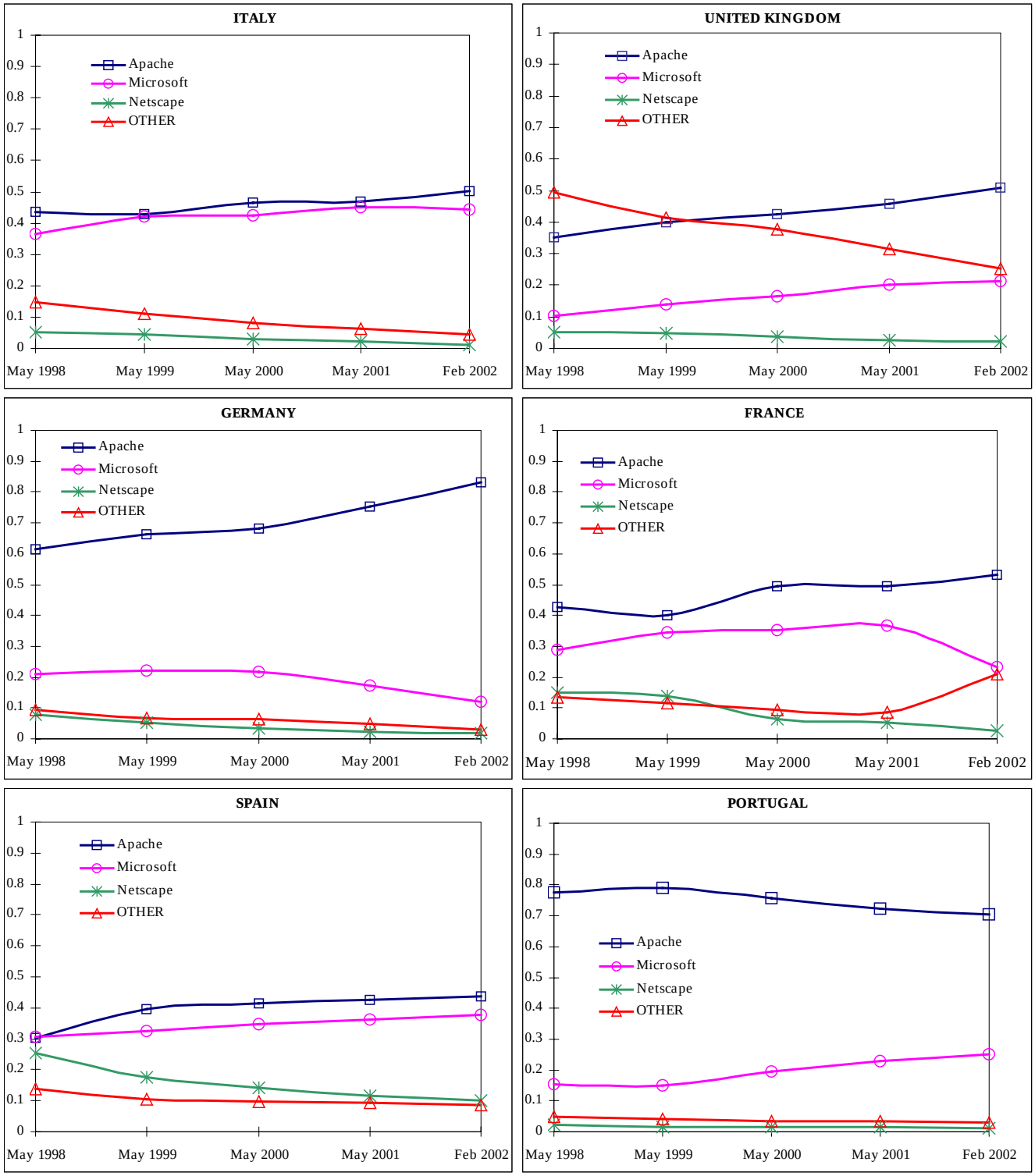
Source: Security Space (2002)

Figure 3.4 Web Server Market Shares in Europe and United States



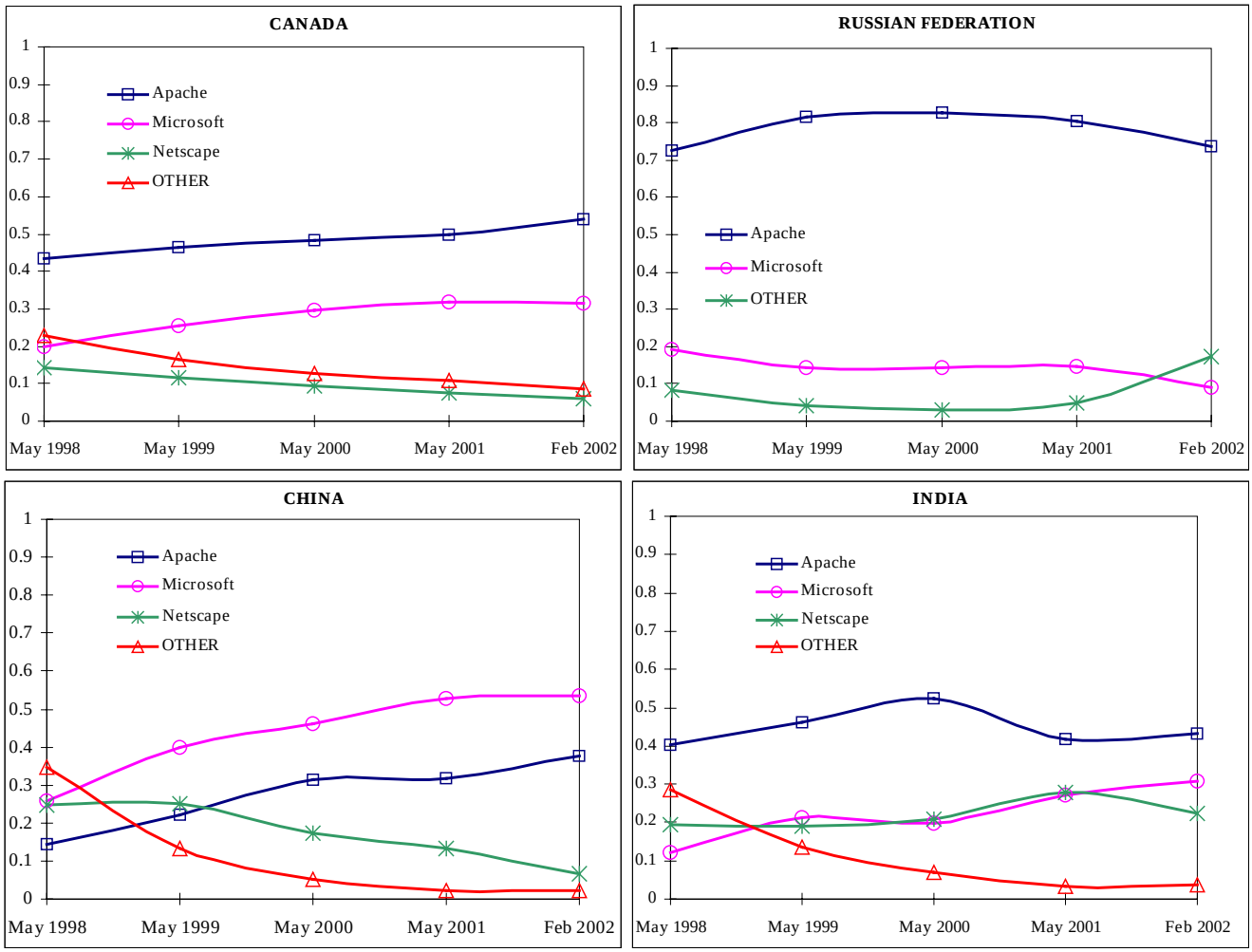
Source: Security Space (2002)

Figure 3.5 The largest EU countries



Source: Security Space (2002)

Figure 3.6 Other countries (May 1998 - Feb 2002)



Source: Security Space (2002)

Server operating systems

Several surveys analyse the adoption of different types of operating systems, by using data on worldwide shipment units and sales.

For example, IDC has analysed the adoption of *server operating environments* in 2000 (IDC, 2000) and exhibited unit shipment data for Windows NT, Windows 2000, Linux, Unix and Novell Netware[17]. Table 3.5 reveals that Windows NT/2000 has the largest share of shipments both in 1999 (38.4%) and in 2000 (40.9%). Linux, the second player with a share of 29.6%, grows faster than Windows in terms of shipments. Other server operating environments show declining shares and negative or constant shipment growth rates.

[17] The IDC methodology is based on IDC models maintained by analysts based on the collection of financial data from public and private companies. In this survey IDC counts new software shipments as new software purchases and full versions upgrades (IDC, 2000).

Table 3.5 Worldwide Server Operating Environments - New Software Licence Shipments by Platform, 1999-2000

	1999		2000		1999-2000
Platform	**Shipments**	**Share (%)**	**Shipments**	**Share (%)**	**Growth of shipments (%)**
Windows NT/Windows 2000	2086	38.4	2508	40.9	20.2
Linux	1322	24.3	1645	26.9	24.4
Novell NetWare 3.x, 4.x, 5.x	1064	19.6	1030	16.8	-3.1
Combined UNIX	826	15.2	826	13.5	0.0
Others	140	2.6	116	1.9	-17.4
Total	**5437**	**100.0**	**6125**	**100.0**	**12.6**

Source: IDC (2000)

In particular, the Unix platforms combined maintained the same number of shipments in 1999 and 2000, although there are relevant differences in the shipments across vendors (.

Table 3.6).

Sun Solaris' shipments show the largest growth rate (40.9%), HP grows at the 20 % rate, while the shipments of IBM and Compaq grew at lower rates (around 10 %). Santa Cruz Operations (SCO) platforms and Others showed a marked decline in shipments, in part to Linux's advantage. The slow growth rate of IBM's shipments could reflect its recent openness towards the Linux platform.

Table 3.6 Worldwide Unix Server Operating Environments New Software License Shipments by Vendor

	1999		2000		1999-2000
Vendor	**Shipments**	**Share (%)**	**Shipments**	**Share (%)**	**Growth (%)**
Sun Solaris/SPARC	186	22.5	262	31.7	40.9
IBM AIX	113	13.7	125	15.1	10.6
HP-UX	105	12.7	126	15.3	20.0
SCO OpenServer	194	23.5	124	15.0	-36.1
SCO UnixWare	119	14.4	93	11.3	-21.8
Compaq Tru64 Unix	31	3.8	34	4.1	9.7
Other Unix Server Operating Environments	78	9.4	62	7.5	-20.5
Total	**826**	**100.0**	**826**	**100.0**	**0.0**

Source: IDC (2000)

The Netcraft Web Server Survey mentioned before (Netcraft, 2002a) provides also data about *operating systems* of web sites and computers on the public Internet. The survey is based on the number of physical computers hosting websites rather than traditional counts of hostnames. To clarify the underlying methodology it is useful to recall that Netcraft sends each month an http request to each

web site in its database, and determines both the web server used to support web sites, and the operating systems through inspection of the TCP/IP characteristics of the response. In detail

> "By arranging for a number of IP addresses to send packets to us near simultaneously, low level TCP/IP characteristics can be used to work out, within an error margin, if those packets originate from the same computer, by checking for similarities in a number of TCP/IP protocol header fields. To build up sufficient certainty that IP addresses on the same computer have been identified, many visits to the sites in the Web Server Survey are necessary, which takes place over a period of over a month" (www.netcraft.com/survey/index-200106.html#computers).

This method can produce some counting errors (under-counting and over-counting) which are reported in the Netcraft methodological notes[18].

The results of the Netcraft survey on the operating systems published in June 2001 (Netcraft, 2001) are in line with those of IDC discussed before. Microsoft Windows is the most widespread operating system with a share of 49.6%, followed by Linux with 29.6% and Solaris with 7.1 % (Table 3.7).

Comparing these data with those on web server Software, it is useful to remind that Windows runs less servers than Apache, which is largely adopted by hosting companies and ISPs usually running large numbers of sites on one or few computers. Windows is most popular with end-users and self-hosted sites, where the number of computer per host is much smaller. Netcraft reports that Linux has gained shares, but not to the detriment of Windows. Instead, other operating systems have lost shares - mostly Solaris and other proprietary operating systems and, and to a smaller degree, BSD operating systems.

Table 3.7 Operating Systems Used by Computers Running Public Internet Web Sites, 2001

Operating System	June 2001	March 2001
Windows (Windows 2000, NT4, NT3, Windows 95, Windows 98)	49.6%	49.2%
Linux	29.6%	28.5%
Solaris (Solaris 2, Solaris 7, Solaris 8)	7.1%	7.6%
BSD (BSDI BSD/OS, FreeBSD, NetBSD, OpenBSD)	6.1%	6.3%
Other Unix (AIX, Compaq Tru64, HP-UX, IRIX, SCO Unix, SunOS 4 and others)	2.2%	2.4%
Other non-Unix (MacOS, NetWare, proprietary IBM OSS)	2.4%	2.5%
Unknown	3.0%	3.6%

Source Netcraft (2001)

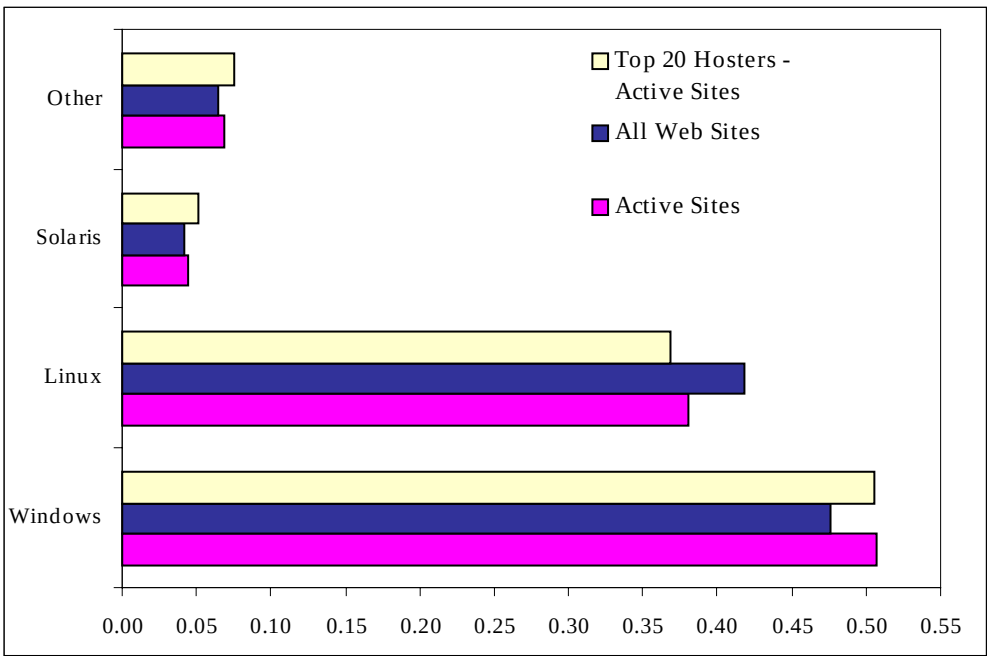
Diffusion of operating systems in Italy

For the Italian market we can report data from the Netcraft Survey which illustrate the adoption of different operating systems amongst the top 20 hosting companies, which account for 63% of the total

[18] For details on Netcraft's methodology for counting computers see http://www.netcraft.com/Survey/index-200106.html#computers.

number of web server operating systems. Figure 3.7 summarises the shares of Windows, Linux, Solaris and other operating systems for respectively all sites, active sites and top 20 hosters. There are not strong differences across these categories, and this is mainly due to the fact that in Italy most web sites are physically hosted by few big hosting companies whose decisions heavily affect the global pattern. Furthermore, Table 3.8 shows the adoption of Windows Linux and Solaris for the 20 Top hosting companies ranked by number of active web sites. Only six out of 20 host companies adopt a single operating system (either Linux or Windows) in more than 90% of their sites, while the remaining host companies' sites adopt two or more operating systems. Solaris is extensively used only by Telecom Italia (20%) and SEAT Pagine Gialle (22%).

Figure 3.7 Web server operating systems, Italy – Feb. 2002



Source: Netcraft (2002b)

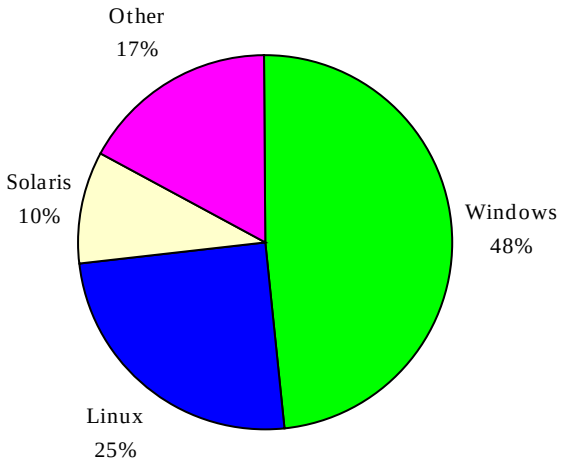
Table 3.8 Operating system adopted by the Top 20 hosting companies, Italy, February 2002.

Parenting Info	Hosting Domain	Total Sites	Windows	Linux	Solaris	Other
Telecom Italia	interbusiness.it	65.989	47%	30%	20%	3%
Aruba.it	aruba.it	64.948	100%	0%	0%	0%
Tiscali	tiscalinet.it	38.578	0%	100%	0%	0%
Infostrada	iunet.it	27.356	30%	15%	0%	54%
DADA SpA	dada.it	20.484	27%	73%	0%	0%
BT Ignite	inet.it	9.578	41%	28%	1%	29%
seeweb.it	seeweb.it	9.571	11%	89%	0%	0%
9NetWeb SpA	9netave.it	8.009	97%	2%	0%	1%
217.73.227	217.73.227	5.304	0%	100%	0%	0%
Telenor	nettuno.it	5.175	49%	31%	4%	16%
Blixer SpA	blixer.it	4.974	66%	34%	1%	0%
Telia	ita.tip.net	4.849	77%	21%	0%	2%
Worldcom	uu.net	4.753	36%	61%	3%	0%
Alicom Italia	alicomitalia.it	3.942	93%	7%	0%	0%
CDC SpA	interfree.it	3.832	5%	95%	0%	0%
MC-Link SpA	mclink.it	3.785	57%	6%	0%	37%
SEAT Pagine Gialle SpA	tin.it	3.673	77%	0%	22%	1%
Cybernet	flashnet.it	3.202	54%	46%	0%	0%
KPN Qwest	comm2000.it	3.000	56%	41%	0%	2%
idc-ti.it	idc-ti.it	2.933	95%	5%	0%	0%
Total for Top 20 Hosters (63% of Italy total)		293.935	51%	37%	5%	8%
Total for Italy (Hosted and Self-Hosted)		468.164	51%	38%	4%	7%

Source: Netcraft (2002b)

Finally, a survey of SSL sites, i.e. sites that use the SSL protocol for encrypted transactions on the Internet, conducted in Italy by Netcraft in February 2002 confirms the lead of Windows, with a 38% market share, followed by 25% of Linux and 10% of Solaris (Figure 3.2).

Figure 3.8. Share of operating systems in SSL sites, Italy.



Source: Netcraft (2002b)

To summarise the main results reported so far it is worth to recall that according to IDC's estimates Windows has the largest share in total shipments while Linux shows the best performance in terms of shipment growth rates. Data collected by Netcraft on web server operating systems shows instead that Linux is the most adopted operating system worldwide, although there are differences in the leadership across countries.

3.4 Operating environments – revenues from open source and proprietary platforms

While in the previous Section we have focused on the number of adopters of different software, in this section we analyse the market shares of different platforms calculated on firms' revenues. It is important to remind that the comparison between open source platforms (Linux) and proprietary one can be biased against the former because open source software licences do not allow license fees and the average price of different distributions are considerably lower than those of the corresponding proprietary software.

The IDC report on Server Operating Environment (2000) shows that in 1999 and 2000 the highest revenues have been obtained with Unix platforms, followed by Windows and Netware. Linux and other operating systems gained much lower revenues. However, Linux, Unix and Windows experienced positive growth rates of revenues from 1999 to 2000 (Table 3.9). Comparing these data with the shipments data, we may observe that although Linux experienced a high share of shipments, the revenues have been quite low. In fact, as exhibited in Table 3.10, the Linux revenue per unit of shipment is the lowest and does not change in the two years. Unix shows the largest revenue per shipment, which grew significantly from 1999 to 2000. Windows and Netware show lower and slightly declining revenue per shipment ratios. Besides the different marketing models mentioned before, these differences in prices across platforms reflect the differences in their target markets, with Unix addressing the medium-high end segment of the market.

Table 3.9 Worldwide Server Operating Environment Revenues by Platform, 1999-2000 ($M)

	1999		2000		1999-2000
Platform	**Revenues**	**Share (%)**	**Revenues**	**Share (%)**	**Growth of revenue (%)**
Combined Unix	3041	56.5	3594	59.9	18.2
Windows NT/Windows 2000	1553	28.9	1719	28.6	10.6
NetWare 3.x, 4.x and 5.X	716	13.3	617	10.3	-13.9
Linux	30	0.6	38	0.6	25.3
Other NOS	40	0.7	35	0.6	-12.5
Total	**5381**	**100.0**	**6002**	**100.0**	**11.5**

Source: IDC (2000)

Table 3.10. Revenue per unit of shipment, 1999-2000 ($M)

	1999	2000
Platform	**Revenue / Shipments**	**Revenue / Shipments**
Combined UNIX	3.68	4.35
Windows NT/Windows 2000	0.74	0.69
NetWare 3.x, 4.x, 5.x	0.67	0.60
Linux	0.02	0.02
Other	0.29	0.30
Total	0.99	0.98

Source: our elaboration from IDC (2000) data.

The IDC 2002 Survey on Integrated Collaborative Environments (ICE) (IDC, 2002a) shows the revenues of ICE by Operating environments. IDC defines Integrated collaborative environments as software that "provide a framework for electronic collaboration, typically within an organization, based on shared directory and messaging platforms. The core functionality areas are e-mail, group calendaring and scheduling, shared folders/databases, threaded discussions, and custom application development" (IDC, 2002a).

Table 3.11 shows that in 2001 Windows had the largest share of ICE revenue, followed by OS/400, Unix and Linux and other open source operating environments. However, for 2002-2006 IDC forecasts negative growth rates for all operating environments but for Linux.

Table 3.11 Worldwide Integrated Collaborative Environments New Software Revenues by Operating Environments, 2000-2006 ($M)

	2000	2001	2002E	2003E	2004E	2005E	2006E	2001 Share (%)	2001-2006 CAGR (%)	2006E Share (%)
Windows 32 and 64	1254	1192	994	853	736	645	594	75.9	-13.0	63.3
OS/400	148	117	98	83	69	59	47	7.4	-16.7	5.0
Unix	120	95	82	72	64	54	48	6.0	-12.7	5.1
Linux/otheropen source	42	55	95	133	164	190	207	3.5	30.5	22.0
Mainframe	36	31	24	21	18	15	13	1.9	-15.5	1.4
Other host/server	99	79	63	50	43	34	28	5.0	-18.5	3.0
Other single user	4	4	3	3	3	2	2	0.2	-8.7	0.3
Total	**1703**	**1571**	**1359**	**1214**	**1095**	**999**	**940**	**100.0**	**-9.8**	**100.0**
Growth (%)	-21.1	-7.7	-13.5	-10.7	-9.8	-8.8	-5.9			

Source: IDC (2002a)

The market for web server software is also an increasingly important source of revenues for software companies. The IDC survey on revenues in the Web Server Software and Web Acceleration

Software (WEBS)[19] market (IDC, 2002b) by operating environments reveals that the revenues of WEBS working in the Unix operating environment are expected to be the largest.

In 2001 66% of revenues from WEBS derived from sales of products running on Unix platform, 28.1 % on the Windows operating environment and lower shares in the other platforms (Table 3.12). However, the market is expected to substantially grow from 2002 to 2006 and Windows, Linux and other open source platforms to grow faster than Unix, by hosting more WEBS and gaining largest market shares. Moreover, the expected growth of low price open source software could limit the future growth of revenues in this market.

Table 3.12 Worldwide Web Server and Web Acceleration Software Revenues by Region and Operating Environment 2000- 2006 ($M)

Operating environment	2000	2001	2002	2003	2004	2005	2006	2001 Share (%)	2001-2006 CAGR (%)	2006 Share (%)
Unix	418.3	566.2	702.1	863.7	993.4	1138.5	1290.4	66.0	17.9	61.4
Windows 32 and	186.9	240.9	313.5	399.1	494.2	612.7	765.9	28.1	26.0	36.4
Mainframe	4.7	5.2	5.4	5.6	5.7	5.7	5.7	0.6	1.6	0.3
Linux/other OSS	4.5	7.6	10.1	12.4	14.9	17.8	21.4	0.9	22.9	1.0
OS/400	2.2	2.5	2.6	2.6	2.7	2.7	2.7	0.3	1.6	0.1
Embedded and	2.2	2.4	3.0	3.8	4.3	5.0	5.8	0.3	19.3	0.3
Other host/server	25.9	30.7	29.1	27.7	19.4	13.6	9.5	3.6	-20.9	0.5
Other single user	1.6	2.0	1.8	1.7	0.8	0.2	-	0.2	NA	-
Total	**646.2**	**857.5**	**1067.6**	**1316.5**	**1535.3**	**1796.2**	**2101.3**	**100.0**	**19.6**	**100.0**
Growth (%)	NA	32.7	24.5	23.3	16.6	17.0	17.0			

Source: IDC (2002b)

3.4.1. The market for Linux in Italy

In the Italian market, the opportunities of revenues from Linux are growing and are expected to grow in the next years. A Survey conducted by Sirmi (2002) reports that the value added from the Linux software have grown at a growth rate of 44% between 1999 and 2000, 98 % from 2000 to 2001, and is expected to keep this pace in the subsequent years. The growth rate of services and software for dedicated environment have been much stronger and are expected to grow faster (see Table 3.13), suggesting that the opportunities for revenues in this market are larger for specific software development and services.

[19] Web server and Web acceleration software (WEBS) is software that "allows systems or nodes to access files stored on a local server, or it can act as a relay station for information stored on other servers in the network" (IDC, 2002b).

Table 3.13 Linux Market in Italy, 1999-2003 (Value added, Millions of Euro)

	1999	2000	2001	2002 E	2003 E	1999-2000 %	2000-2001 %	2001-2002E %	2002-2003E %
Linux	7.7	11.1	22	40.2	68	44%	98%	83%	69%
SW for dedicated environment	-	1.8	3.6	8.2	19.3	-	100%	128%	135%
Total SW	7.7	12.9	25.6	48.4	87.3	68%	98%	89%	80%
Services	43.8	92.9	205	438	851	112%	121%	114%	94%
Total Linux	51.5	105.8	230.6	486.4	938.3	105%	118%	111%	93%
Number of Servers	9100	12300	19800	35000	53000	35%	61%	77%	51%

Source: Sirmi (2002)

This pattern is also confirmed by the analysis of the revenues of the main Linux distributors (including two IBM partners) (see Table 3.14). It is apparent the effort of IBM to support Linux through partnership with the main Linux distributors such as Red Hat and Suse. The revenues growth rates of RedHat, Suse and Caldera have been impressively large, and the share of revenues accounted for by services is substantial (Sirmi, 2001a).

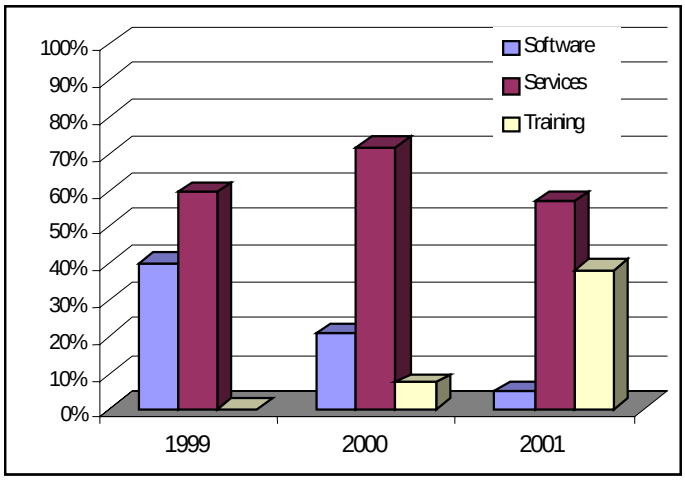
Another survey on the Linux vendors in Italy (Sirmi, 2001b) also shows that most revenues come from services and an increasing share derives from training activities, while the revenues from software are declining (Figure 3.9).

Table 3.14 Linux distributors' revenues– Millions of Euros

	2000	2001	2000-2001 (%)
RedHat*	8.76	21.9	250 %
Software	-	10	
Services	-	11.9	
SuSe*	3.2	12.4	388 %
Software	-	6.2	
Services	-	6.2	
Caldera	2.06	3.5	170 %

*IBM partners
Source: Sirmi (2001a)

Figure 3.9 Linux Vendors' Revenues, Breakdown by Activity 1999-2001



Source: Sirmi (2001b)

3.5 The market for e-mail servers

There are not many data available on the adoption of other application software. Among the few publicly available data there are those arising from a survey conducted by D.J. Bernstein between September 27th 2001 and October 3rd 2001 on the market for email server software (Wheeler, 2002). The survey is based on over one million of random IP addresses and shows that Sendmail is the leading email server worldwide (Table 3.15). Microsoft Exchange and Unix qmail are the other most popular products. Among the products reported in Table 3.15, Sendmail, Unix Postfix and Unix Exim are open source software and cover altogether 46 % of the market.

Table 3.15 Email Server Market Shares

EMAIL SERVER(*)	Market Shares
Sendmail	42%
Microsoft Exchange	18%
Unix qmail	17%
Windows Ipswitch Imail	6%
Unix smap	2%
Unix Postfix (Vmailer)	2%
Unix Exim	2%
Others	1%

(*) Sendmail, Unix Postfix and Unix Exim are OSS products.
Source: Bernstein Survey on random IP addresses (27 September 2001 - 3 October 2001).

3.6 The demand for open source software

There are not many studies available on the international demand for open source software.

In Italy a survey from Sirmi (2001b) on the Linux market reported that the target market for the Linux vendors are large customers (like Daimler-Chrisler, BMW, Iveco, AEG, Banca d'Italia, IlSole24Ore, SAI Assicurazioni, Sogei, Fiscali), ISP and ASP, and future efforts will be made to target the SMEs (Small and Medium Enterprises). Ad hoc surveys can be developed to assess the demand for open source software and the comparison with commercial proprietary software.

3.6.1. The academic market

Specific surveys have been carried out to analyse the adoption of open source and proprietary operating system software in the academic market.

In April 1999, the *Internet Operating System Counter* (Zobelein, 1999) analysed all servers (running at least http, ftp, and news services) on '.edu' domains within the RIPE NCC network in Europe, Middle East, North of Africa and parts of Asia (see http://www.ripe.net/ for more details), counted by

domain name. The survey shows that Windows is the first operating system in '.edu' domains on the public Internet (33.4%) followed by Solaris (22.4) and Linux (10.8 %).

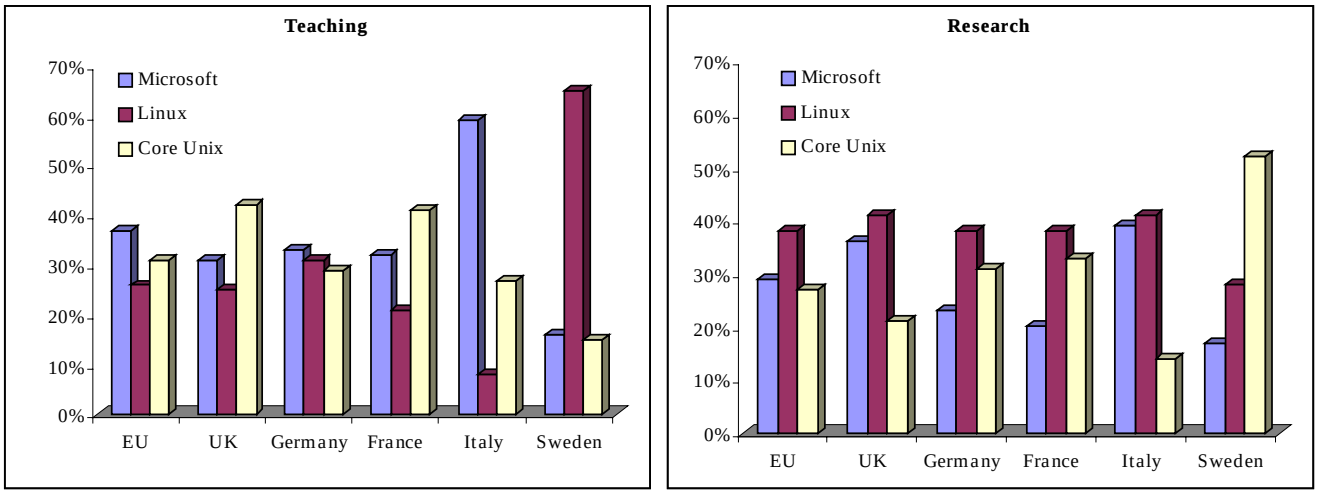
Table 3.16 Operating Systems in the academic market, 1999

Operating System	% adoption
Windows95/98/NT	33.4
Solaris/SunOS	22.4
Linux	10.8
Mac/Apple	8.8
IRIX	4
BSD Family	3.5
AIX	3.2
HPUX	2.8
DigitalUnix	2.8
NovellNetware	1.9
ReliantUnix/Sinix	0.8
SCOUnix	0.3
Others	5.3
Total	**100**

Source: The Internet Operating System Counter, www.leb.net/hzo/ioscount

Total Romtec (2001) carried out a survey on the university market for operating systems in four European countries (Italy, France, Germany, UK and Sweden), through 588 face-to-face interviews with research and teaching staff and PhD students in Computer Science and Computer related Departments. They found that Microsoft operating systems are the most used for teaching purposes (37 %) while Linux is the most used for research activities (38%) (Figure 3.10). However, there are differences across countries. For teaching, the leadership of Microsoft is very marked in Italy (59%), while in France and the UK Core Unix is the most adopted operating system and in Sweden the most used is Linux (65%). In the research field Linux is the most adopted platform in all countries, except for Sweden where Core Unix is the most used. In Italy the share of Microsoft OS is significant and very close to that of Linux (respectively 39 % and 41 %).

Figure 3.10 Primary Operating Systems Used in Teaching and Research, 2001



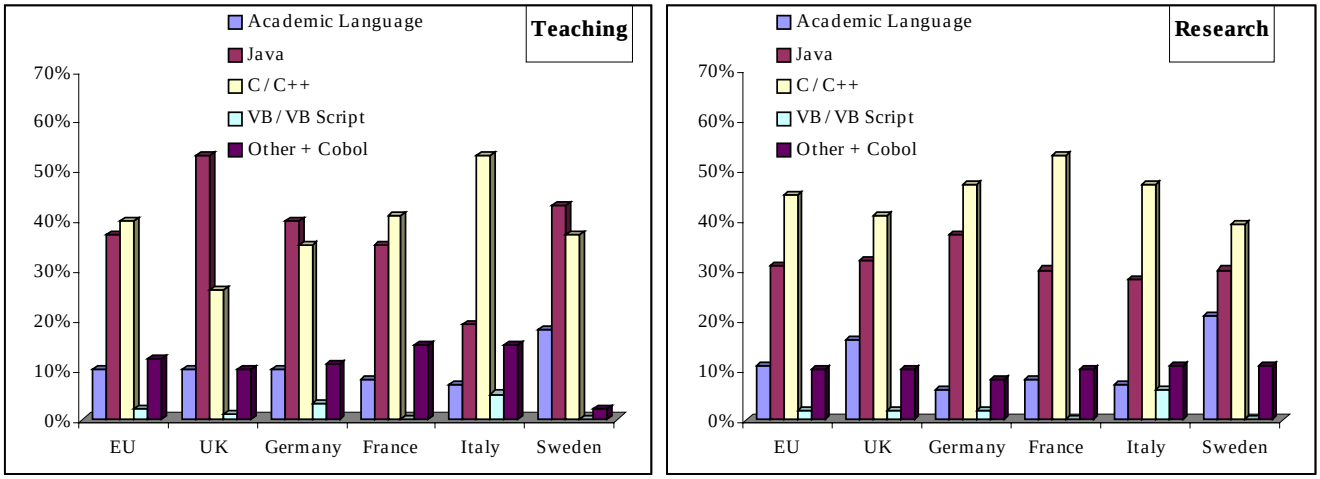
Source: Exec deck Total Romtec, October 2001
% of respondents using of a primary OS

The Total Romtec Survey also analysed the adoption of programming languages in teaching and research in European universities and found that C/C++ and Java are the most used language for teaching. As far as teaching is concerned, C/C++ is the most preferred language in France and Italy while Java is most popular in the UK and Sweden (Figure 3.11). For research activities, C++ is always the most used language, followed by Java.

The survey also found that GNU C++ is the primary used C/C++ tool, and also the most generally used (when multiple responses are allowed), followed by Microsoft as primary used and by Emacs as used language (Table 3.17). As far as Java tools are concerned, Sun JDK is the most used, followed by Emacs (Table 3.18).

It is worth to note that the tools and languages mostly used in the development of open source projects (i.e. Java, GNU C++, Emacs) are also very extensively adopted by Universities.

Figure 3.11 Primary Language Used in Teaching and Research, 2001



Source: Exec deck Total Romtec, October 2001

Table 3.17 C/C++ tools Used in Teaching and Research - Europe 2001 (% of respondents using)

	Primary	Used*
MS Visual C/C++	19%	39%
C/C++ tools other than MS (inc. 1 & 2)	67%	89%
GNU C++ (1)	*37%*	*60%*
Unix C++ (2)	*14%*	*38%*
Emacs	10%	46%
Generic tools for C/C++ other than MS	4%	49%
Total	100 %	-

Source: Exec deck Total Romtec, October 2001
* multiple answers

Table 3.18 Java tools Used in Teaching and Research - Europe 2001 (% of respondents using)

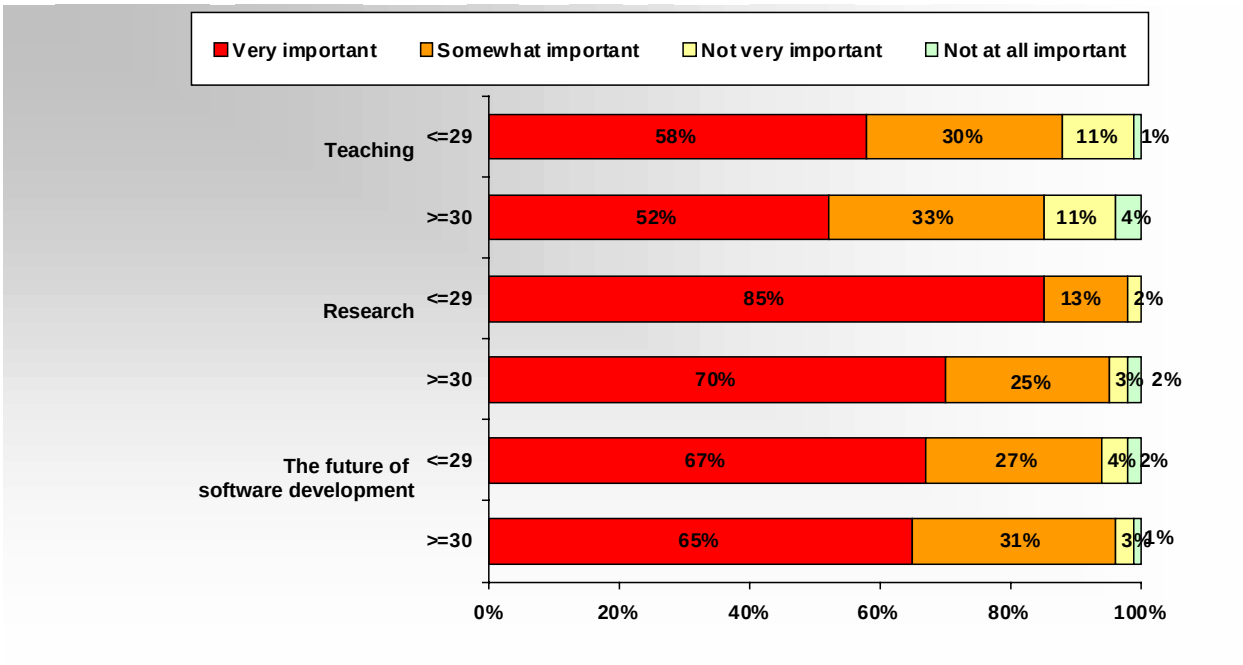
	Primary	Used*
MS Visual J++	8%	16%
J/J++ tools other than MS (inc. 1 & 2)	67%	88%
Sun JDK (1)	*43%*	*68%*
IBM Visual Age for Java (2)	*4%*	*10%*
Emacs	16%	37%
Generic tools for J++ other than MS	9%	31%
Total	100 %	-

Source: Exec deck Total Romtec, October 2001
* multiple answers

Finally, the survey analyses the type of licence used in the development of software and the importance of access to the source code for teaching and research, and for the future of software development. The survey reports that when software is developed for research aiming at the publication of results, for teaching purposes or for research funded by the government or by the European Union, it is mostly left on the public domain (no license is introduced). GNU/GPL licenses and copyright are other most common strategies. When the software is developed for research funded by the industry, the copyright is retained in most cases.

Although in some cases the scientists prefer to retain the copyright over their software, the access to the source code is considered very important especially for research and for the future of the software development (see Figure 3.12).

Figure 3.12. Importance of access to source code



Source: Exec deck Total Romtec, October 2001

4. License models: a comparison between proprietary and open software

4.1 Introduction: Intellectual Property Rights and software

In the software industry the appropriability of innovation, including intellectual property rights (IPR), is quite weak. This is in part due to the relatively young age of this industry and in part to the characteristics of software technology.

Since the unbundling of software from hardware in 1969, which gave rise to the birth of an independent software industry, the issue of IPR in software has become the object of a lively debate between the advocates of a strong legal protection, which point out the importance of incentives to innovation, and the advocates of a weak protection, which highlight the social benefits of high entry rates, competition among different technological standards and diffusion of technological knowledge. In line with the supporters of a weak legal protection there is a stream of the literature which makes the point that in industries characterised by strong network externalities, such as operating systems, where *competition between alternative standards* is structurally weak, IPR should be weak to guarantee entry and competition among complementary technologies *within the standard* (e.g., application software) (Merges, 1996; Cohen and Lemley, 2001). The advocates of weak IPR claim that a strong IPR is not a necessary condition to guarantee a rapid rate of change in this industry since many important software innovations have been introduced before a strong IPR have emerged (see Merges, 1996, for a wider illustration of this point). On the other hand, they claim that a strong IPR may have negative effects on the rate of technical change since many important innovations in this industry have been introduced by new firms rather than established companies (Prusa and Schmitz, 1991).

To our knowledge, this debate has not yielded so far any clear-cut conclusions as to the relationship between IPR regime and rate of innovation in this industry. A deeper exploration of this issue however should be addressed in future research.

To our purposes here, we begin by introducing very briefly the two traditional IPR systems – copyright and patents. In the following sections we shall turn our attention to copyright in the context of proprietary software and in the open source software community.

Until recently software has been considered as an intellectual activity whose products fall into the 'artistic' sphere and as such they should be protected by the copyright law. This law provides the author with the rights to use, copy, modify, and distribute the code. It does not impose any obligation about the disclosure of the source code together with the object code. Copyright holders typically sell the right to use or modify the software to third parties through a license contract.

Like other mathematical formulae and mental processes, software inventions have not been considered eligible for patent protection until recently[20]. During the 1980s and the 1990s the US courts developed new doctrines that progressively recognised software as 'useful art' and then granted software inventions patent protection while Europe kept a more conservative position. During these two decades the US Patents and Trademarks Office (USPTO) has granted about 80,000 patents to software-related inventions (Cohen and Lemley, 2001). The US courts officially recognise software patentability for the first time in 1981. In the *Diamond v. Diehr* dispute[21] the US Supreme Court allowed software patents provided that they were used for running a machine or an industrial process. This decision of the Court gave rise to 'the doctrine of the magic words' since after that decision a lot of software patents were granted to software inventions associated to any physical element, even though the only element of novelty was in software rather than hardware - 'nearly any physical element or step would suffice to render statutory a claim that recited' the magic world that linked the unpatentable software to 'otherwise statutory process or apparatus' (Cohen and Lemley, 2001, p. 9). Since 1994 software per se has been admitted as patentable provided that the claims explicitly referred to 'computer programs implemented in a machine' and the association with any physical devices was definitely abandoned in 1998, when the Federal Circuit recognised that a "transformation of data" producing "a useful, concrete ad tangible result" represents a practical application and is then patentable[22].

However, even before 1994 patents have been granted not only to software inventions bundled with physical apparatus, but also to pure software in the form of data structure, data compression, encryption algorithms, data processor for calculus. This happened because of some peculiarities of the software technology and the institutions that govern the patent system: 1) until 1995 the Patent Office did not hired any computer scientists among its patent examiners; 2) due to the 'doctrine of the magic word' most software patents were classified in the field of physical applications (e.g., pizza ovens and

---

[20] Courts declared mathematical formulae and algorithms as non patentable subject matter in the Court case *Gottschalk v. Benson* in 1972, 409 U.S. 63 (Cohen and Lemley, 2001).

[21] 450 U.S. 175, 1981 (Cohen and Lemley, 2001).

[22] Appalat Federal Circuit Decision F.3d 1526 (Fed. Cir. 1994) and the State Street Bank & Trust Federal Circuit Decision 149 F.3d 1368 (Fed. Cir. 1998) (Cohen and Lemley, 2001). In Europe, the European Patent Convention signed in 1973 states that computer programs 'as such' are not useful inventions and therefore cannot be patented (art. 52). However, a recent of EU directive to the European Parliament has recognised the most recent doctrine developed by the European Patent Office Board of Appeal). According to the proposal computer programs have by definition a technical dimension and can be considered as patentable inventions. In line with the patent law, a software invention has to provide a technical contribution in a technical field, that is advancement of the "state of the art" which should result non obvious to a "skilled practitioner". Unlike the US judicial practice, then the EU law is oriented to maintain a formal link between software and hardware since software inventions can be protected as products (i.e., a programmed computer or computers network) or as processes (i.e., a sequence of operations made possible by the execution of a software program). The EU proposed directive then excludes the protection of software programs per se. Non technical inventions, like commercial methods developed, for instance, to run e-commerce services, are not patentable in the EU while they have been granted the status of patentable invention in the US.

other machines) rather than in computer science fields; and this made it difficult to keep record of the prior art which is important to establish patent infringements.

Even today, however, the application of the standard patent law to software is problematic. Besides the issues discussed before, it is worth to remind that technical progress in software is characterised by a flow of incremental, modular and sequential innovations. More than in other technical fields, initial software inventions are rapidly followed by many generations of subsequent incremental inventions that, taken all together, may depart significantly from the initial invention. Along with modularity and interoperability, incremental technical change gives rise to interdependences across different generations of sequential innovations and therefore increases the probability of infringement litigations. Another peculiarity of software is represented by the lack of formal systems of scientific and technical documentation like publications in journals. Unlike other more established engineering fields, most software inventions are described by the source code, which is kept secret by inventors (Cohen and Lemley, 2001).[23] It is important to note that the patent law does not impose specific disclose obligations over software inventors. But even if the patent law were modified to impose the disclosure of source code this would not be enough to guarantee a full disclosure of information that is a key principle of the patent system in general. To understand the whole structure of a software programme a 'skilled practitioner' has to gain access to the entire source code of the programme; however, the patented code is normally only a subcomponent of the whole programme. But, unlike the copyright law, which allows reverse engineering for specific purposes ('fair use doctrine'), the patent law precludes the possibility of reverse engineering (either black box reverse engineering and decompilation) because reverse engineering (by decompilation) would require making a temporary copy of the programme in RAM memory. For these reasons, some are in favour of a narrow patent scope for software (e.g., Merges, 1996; Cohen and Lemley, 2001). For example, Cohen and Lemley (2001) argue that the application of current patent law to software should take into account the peculiarities of this industry, such as sequential innovations and the lack of documented prior art. In light of these peculiarities a broad patent scope in software would introduce significant distortions in the market for software technology. To minimise the negative effects of the patent system Cohen and Lemley propose: i) the courts (and/or the legislator) allow a limited right to reverse engineer patented software in order to allow 'inventing around' existing software inventions- i.e., to gain access and study unprotected elements of patented programs, to duplicate these elements and make improvements; ii) that the courts apply the 'doctrine of equivalents' narrowly in infringements cases. This doctrine aims to establish the existence of substantial equivalence between the elements of a patented program and those of an allegedly infringing program.

If the elements of the two programs appear to be interchangeable to a 'skilled practitioner' at the time of the alleged infringement then the accused program does not pass the test of equivalence. Cohen and Lemley suggest that the courts should refuse a finding of equivalence if the accused element is 'interchangeable' with prior art that should have narrowed the original patent or if the infringing improvement is several generations away from the initial invention (p. 4).

Within the current legal system, copyright, patents and trade secrets are complement both in the US and in Europe in that the same software invention can be protected under these three laws. In particular, the absence (copyright) or weak disclosure obligations (patents) make it possible to cover under trade secrets the source code and to license the object, executable code against the payment of license fees.

4.2 Copyright and licensing open source and proprietary software

The market for software is influenced by the property rights regime, which affects the scope of property rights (from patents to copyleft), and the contractual regime, which provides the institutional framework within which software technology can be transferred across individuals and organisations (e.g., by different licensing arrangements).

OSS is different from proprietary software on both grounds – the property right regime and the contractual regime. These differences are reflected in the organisation of marketing and distribution of software products and services.

In this section we focus on different contractual arrangements (license models) that are adopted by proprietary and open source software producers to distribute products and services.

Proprietary commercial software is distributed under licenses that usually limit the use, and deny the possibility to copy, modify and distribute (distribution is prohibited unless the distributor pays royalties to the copyrights holder). Furthermore, commercial proprietary software is usually protected by trade secret. In this way commercial software companies can extract rents from their R&D activity, also protecting their innovation from competitors. Unlike pure information, proprietary software is then produced and distributed as a private good.

In licenses that accompany open source software, the copyrights holder maintains the right to use, modify and distribute the software but gives up the trade secret over the source code and allow users the same rights, therefore making the software a public good.

---

[23] The alleged lack of information published in scientific and technical journal is something that should be proved. As a matter of fact, there are important international journal such as the IEEE magazine, which provide access to software

Open Source Software has stimulated significant contractual innovations that are documented by the proliferation of licensing schemes. In general terms, the licensing schemes define the degree of protection of the source code, the ways in which the code can be used or distributed, the level of enforcement of the code protection. Appendix A presents the list of software licences drawn from the Open Source Initiative web site[24]. In sections 4.2 and 4.3 we group these licences into few broad categories which share similar characteristics.

We describe four different types of software distribution (proprietary software, public domain software, Open Source/Free software, Shared Source software) and compare them according to several characteristics of the adopted licence models (Table 4.1).

'Proprietary' software

Proprietary software is software that is usually distributed in object, machine-readable (binary) code. The copyright holder does not provide the user with the right to copy, modify and distribute the software (this software is also referred to as 'closed' software by the advocated of open source software). It is also possible that proprietary software is distributed with the source code, but the rights of copying, modifying or distributing the source code are usually not provided to the users. However, it is not unusual that users of (proprietary) custom or bespoke software obtain the source code along with the object code. Proprietary software includes three categories of software: *proprietary commercial software*, *shareware*, and *freeware*.

> *Proprietary commercial software* is a closed software distributed in the form of packages in exchange for a fixed fee (or a royalty) or bespoke programmes (e.g., applications that meet the specific requirement of a customer).
>
> *Shareware* is usually distributed on a trial basis with the agreement that the user can buy it or extended version of it later on. For example, software developers may offer a shareware version of their program with a built-in expiration date so that users have to underwrite a right to use license in order to use the software afterwards (e.g., Symantec's Norton Antivirus). Other shareware is offered for free but with some functionality disabled, while the full version is available for a fee. Users can redistribute copies but subsequent users have to pay a license fee to the holder of the copyright.

---

scientific and technical advances .

Freeware is software that is offered in object form at no usage cost (no license fees). However, only private use is allowed of the software. Licence agreements characterising freeware permit free copy and distribution only for non commercial purposes (i.e., private use or training purposes). In general, freeware licences claim that the 'underwriter' may use, copy and distribute - for non commercial purposes - the executable software, provided that any copy contains all the original software copyright, trademark and other proprietary notices. However, the 'underwriter' cannot i) modify, translate, reverse engineer, disassemble or otherwise attempt to reconstruct or discover the source code of the software (except to the extent to which applicable laws specifically prohibit such restrictions); ii) create derivative works based on the licensed software; iii) rent, lease, loan or resell the software. [25]

Free Software/Open Source..

Free or Open software is software whose source code is distributed together with the object, binary software. It is distributed under license contracts that allow anyone (developer, user) to download the source code, make modifications (further development of lines of code, debugging, etc.) and redistribute it either for free or for a fee. The difference between free software and open source software is to a large extent an issue of emphasis on the constraints to users to limit the distribution of a piece of free code. The meaning of free software today is associated to the Free Software Foundation, founded by Richard Stallman in 1983, and the GNU General Public License (GNU/GPL) which establishes very strong users' rights (Stallman, 1999). As a matter of fact, GPL allows users to have access to the source code on condition that he/she will make the source code freely available (or at the distribution costs which tend to be zero with the Internet) to third parties (copyleft property).[26] Therefore, the underwriter can distribute the object code for a fee but cannot charge a price for the source code that must remain free. The term 'open source' was introduced during a meeting held in February 1998 in Palo Alto, California by 'hackers' that aimed at reacting to Netscape's announcement that planned to give away the source of its browser. Among the participants to this event there was Eric S. Raymond that had been invited out by Netscape to help them to plan

---

[24] See http://www.opensource.org/ for information about OSI and OSI approved licences. See also http://www.gnu.org/licenses/licenses.html for a complete list of licences approved by Free Software Foundation (FSF).

[25] An example of freeware is Ipswitch WS_ftp. Its license claims that "Ipswitch grants you a non-exclusive license to use the Software free of charge if a) you are a student, faculty member or staff member of an educational institution (K-12, junior college, college or university); b) you are a United States federal, state or local government employee; or c) your use of the Software is exclusively at home for non-commercial purposes. Government contractors are not considered government employees for the purposes of this Agreement."

[26] It is important to remind the FSF first copyright the software and then add the distribution terms described above with the aim of protecting the 'freedom' of users.

the release and subsequent actions. After the announcement a part of free software developers realised that it was possible to build a strategy to show to the business world the superiority of an open development process. It was decided to substitute the term 'free software', often associated to an idealistic position against commercial software, with the new label 'open source' contributed by Chris Peterson (www.opensource.org/history.html). When the Open Source Initiative (OSI) was created in February 1998, the Open Source Definition was derived from the Debian Free Software Guidelines edited by Bruce Perens in 1997, removing Debian's specific references (www.opensource.org/definition.html). The OSI approach to users' rights and intellectual property rights is more pragmatic than that of the FSF. To put is simply, underwriters of GNU/GPL licences sponsored by the FSF accept the idea underlying the 'viral clause' according to which any derivative work of a program licensed under the GPL has to be distributed under the same licence scheme. This applies to "any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof". The Open Source Initiative allows license arrangements that give up the 'viral clause' and therefore permits that combinations of proprietary and non proprietary software are distributed in bundles[27]. According to the OSI this approach aims to establish an easier ground for marketing free software overall (www.opensource.org). Neither OSI nor FSF however are against commercialisation of software since they share the view that what matters to guarantee user's freedom is the access to the source code of the program, not the cost of the (object) code that can be charged to the user (hence free or open software as 'free speech and not free beer', see www.gnu.org/philosophy.html). Even if we do not explicitly distinguish between free and open source software, the differences between these two approaches are reflected in the license models analysed here.

Often the terms *freeware* and *free software* are confused. Richard Stallman has clarified that "free it's for freedom and not for price" (see http: // www.gnu.org / philosophy / categories.html for further details). The term 'free' software refers to the right of using, modifying and distributing the source code, not to the possibility of getting the software without paying. By contrast, freeware can be freely downloaded without paying a fee but the source code is not supplied and reverse engineering is not allowed

[27] The OSI claims that only software linked with GPL-ed libraries forming a single work inherits the GPL, not any software which is distributed with GPL-ed Libraries (www.opensource.org/docs/definition.html).

Shared Source code

This is software distributed both in the form of object and source code in order to allow particular customers, partners, and developers (such as big companies, public administration, and university) to solve problems or to adapt the software to specific needs. For example, Microsoft offers different license agreements to specific customers allowing to analyse and reference source code, adapt it to specific purposes such as enhancing system's security, customise the software for high-end customers, adapt hardware, and research-didactical uses. However, users generally have not the right to modify and distribute the software except for licensing programs applied to universities and other research organisations[28]. According to Microsoft[29] this new licensing policy is flexible enough to guarantee the satisfaction of specific users' needs. In addition, it improves feedbacks and debugging while maintaining the intellectual property rights "needed to support a strong software business".

- *Public domain software* is software for which the developer has given up all his/her copyright (or these rights have expired). So there is no 'ownership' over the software that can be freely copied and distributed. Since public domain software is available in the form of source code, it can be modified, packaged and sold at a price above zero. It is one of the most popular way for distributing software in the scientific community.

In this section we compare these four types of software distribution according to the following characteristics of the license contracts :

- *Source code availability*: availability of the source code together with the object code;
- *Underwriter's patrimonial rights*: right to use, copy, modify and distribute the original code;
- *Underwriter's moral rights protection*: degree of protection of moral rights like the acknowledgement of authorship;

---

[28]More precisely, Microsoft provides license agreements that allow business enterprises, system integrators, governments, and OEMs (Original Equipment Manufacturers) to analyse and reference the source code of Microsoft platforms (Windows 2000, Windows XP, and Windows.Net Server) under a non-disclosure agreement; however, the underwriters of these particular licenses cannot modify the source code. Microsoft Research Source Licensing Program authorises faculty, staff and students to use, reproduce, and modify Windows platform source code for "educational purposes and sponsored government and commercial research" (www.microsoft.com/licensing/sharedsource/licensing); the agreement may be extended also to organisations affiliated with a university research centre. Moreover, Windows CE Shared Source Licenses Program "allows the user to access more than 1.5 million lines of Windows CE source code for any non-commercial purposes, including distributing derivative works" (www.microsoft.com/licensing/sharedsource/licensing). Further details on Shared Source Licensing agreement, including C#/Jscript/CLI implementation Shared Source Licensing and Windows CE .Net Shared Source Academic Curriculum License, are available at www.microsoft.com/licensing/sharedsource/licensing.

[29] "The Microsoft Shared Source Philosophy" at www.microsoft.com/licensing/sharedsource/philosophy.asp/ and "The Commercial Software Model", May 3 2001, at www.microsoft.com/presspass/exec/craig/05-03sharedsource.asp/.

- *Initial developer's 'special' rights*. We check whether individuals or organisations that hold the copyright have any right that people whom the software is distributed do not have;
- *Is the software under the licence gratis?* We analyse whether the software is available without paying a fee and whether people who receive the rights from the copyright holder can apply a fee or sell software packages based on the covered code;
- *Provisions for warranty, liability, responsibility for claims.* We analyse the responsibility towards users and third parties (e.g., holders of property rights over software distributed under traditional license arrangements).
- *Provisions for third party claims*. We analyse whether the licence contains explicit rules for third party claims;
- *Compatibility with National Laws*. We analyse whether the rules established by the licence are in conflict or not with national laws; we also check whether each rule contained in the license can be enforced without problems in any jurisdictions;
- *Litigations/Conflicts/Violations*. We analyse whether the licence contains explicit rules for litigations, conflicts or violations.

Table 4.1 summarises the above characteristics for the four types of software distribution and highlights different levels of attribution and protection of patrimonial and moral rights in Open Source and proprietary software distributions. At this stage of our analysis we compare different types of distribution with a 'representative' OSS licence which takes into account the common features of different OSS licenses. In the next section we enter into the details of the most popular OSS models.

Some general issues emerge from the comparison of these licenses.

First, while all OSS licenses overall guarantee the source code availability, proprietary software licences generally do not allow the access to the source code. This restriction is an important constraint especially for sophisticated users (like scientists) or for users who need the source code to develop custom software. Shared Source licences meets in part these categories of users and represent an exception among proprietary software.

Second, proprietary licences do not provide the right to copy, modify and distribute the software, and impose several rules on the use. For example, they often include limitations on the number of machines on which the software can be installed and the location of the machines. By contrast, OSS licenses ensure that users are free to use and copy the software without limitations. However, as we shall see later on, there are also significant difference across OSS licences, with the GPL being the most rigid in protecting the 'freedom' of users to see and modify the original source code even when free

software is distributed in combination with 'non free' software. The implications of the 'copyleft property' and the 'viral clause' mentioned before will be analysed at length in the next section.

Third, proprietary licenses seem to fit better with national laws than OSS in general. For example, the 'viral' clause of the GPL can conflict with the copyright law when the author of modifications or integration to the original covered code can demonstrate that those modifications are original intellectual works which rely on the original GPL covered code only to a limited extent. Moreover, the GPL does not contain explicit export rules and can conflict with restrictions on exporting particular kinds of software – such as source code for encryption algorithms - to certain countries. On the other hand, license contracts for proprietary software are usually written for the purposes of specific customers and countries and even when this is not the case (e.g., with widespread software packages such as Office or SAP) the standard license contains explicit rules for making it compatible with national laws (for example specific rules about limitation of liability and the possibility to do reverse engineering) of the country in which the package is distributed. For example, the Microsoft Windows 98 license claims that if the software is bought in an European country it is forbidden to convert, de-codify, de-compile, and disassemble the code beyond the purposes allowed by national laws.

Proprietary licenses written for specific categories of customers and different purposes may contain explicit rules for solving third party claims and litigations, conflicts and violations.

Finally it is important to remark that, even though OSS generally is not against commercial software, there are significant differences between proprietary and OSS licenses. Users typically do not buy a proprietary software, they only acquire the *right-to-use* against the payment of a royalty or a fee. Therefore proprietary software is 'owned' by the original developer who 'holds' the intellectual property protected by patents and/or copyright.[30] The level of the right-to-use fees is a function of the perceived value of the software and the competition in the market for that particular software.

In the OSS community original developers possess the original code and as such they are holder of copyright to it. As Eric Raymond has recently claimed "the owner of a software project is the person who has the exclusive right, recognised by the community at large, to *distribute modified versions*" (emphasis in the original) (Raymond, 2001, p. 71). How this right is used by the holder depends on the characteristics of license contracts. In general, in the OSS context users are free to use, copy, modify, and distribute software without paying any royalty but he/she may be asked to pay a fee in order to download the code, or may buy directly a low price package containing the OSS software. Even the GPL, which is the most extreme type of OSS license model, explicitly allows to charge a fee for the physical act of transferring a copy of the source, and allows to offer warranty protection in exchange

[30] An exception is represented by bespoke software whose property is often transferred to customers.

for a fee. At the same time the possibility to sale the software in executable form is not explicitly excluded but no (right-to-use) license fee is allowed under this licence scheme (see Hecker, 2000, p. 4).

Table 4.1 A comparison of different software distribution models

LICENCE	'Proprietary' software	Shared Source	Free/Open Software	Public Domain Software
Product Examples	**Commercial**: Microsoft Office **Shareware** WinZip, Norton Antivirus (trial version) **Freeware** Microsoft Internet Explorer, Eudora Email 5.0, Tcl	Windows 2000, XP, .Net Server for Enterprise, System Integrators, Governement, OEM and Research Organisations; C#/Jscript/CLI implementation; Windows CE .Net	Gnu-Linux, Red Hat 7.2, SuSe, AA-Lib., PHPLib, Netscape Navigator, Xfree86	SQLite, PHP pdf creation, CMU Common Lisp
Source Code Availability	No	Yes, conditioned	Yes	Yes
Right to Use	Yes, conditioned	Yes conditioned	Yes	Yes
Right to Copy	Not generally; **Freeware** licences allow copying for non-profit purposes	Yes, conditioned. Microsoft Research Source Licensing Program gives the right to *reproduce* source code.	Yes	Yes
Right to Modify	No	Yes, conditioned. Microsoft Research Source Licensing Program gives the right to modify source code.	Yes	Yes
Right to Distribute	Not generally; **Freeware** licences allow distributions for non.profit purposes	Yes, Windows CE .Net Shared Source Academic Curriculum License admits distribution for non-profit purposes.	Yes, conditioned	Yes
Moral Rights Protection	Yes	Yes	Yes	No
Initial Developer 'special' rights	Yes	Yes	Yes (No in the case of BSD)	No
IS the software under this licence distributed for free?	In general it is NOT for free. The right to use is transferred against the payment of licensing fees. Only **freeware** is always available for free	No, the right to deal with the source code is usually transferred against the payment of a fee	Generally MAY BE *gratis* (no right-to- use license fees), but a fee can bee charged for distributing copies of the software	Yes
Warranty/Liability/ Claims	Total disclaimer of warranty; limitations on liability	Total disclaimer of warranty; limitations on liability	Total disclaimer of warranty; limitations on liability	No explicit rules
Third Party Claims	Explicit rules depending on contractual agreement among parties	Explicit rules depending on contractual agreement among parties	From total disclaimer of responsibility (GPL, LGPL) to absence of explicit rules (BSD)	No explicit rules
National Laws	Total compatibility	Total compatibility	Problems of compatibility could arise (GPL, LGPL)	Total compatibility
Litigations/Conflicts/ Violations	Explicit rules depending on contractual agreement among parties	Explicit rules depending on contractual agreement among parties	No explicit rules except for MPL/NPL	No explicit rules

4.3 An overview of the principal OSS Licence models

All OSS licences share a set of fundamental characteristics. First, they specify if an individual or a company producing the code has the copyright on it. Second, they contain rules that provide the users with a list of rights (to use, modify, and distribute the software) and obligations (i.e., citation of the author in each of the following distribution). Third, the copyright holder often introduces rules in order to guarantee that any work derived from the original product is also open source. Despite these common features, OSS licence schemes differ with respect to other basic characteristics such as the degree of source code protection from *free riding* behaviour, the complying with commercial business models and with national and international laws.

This section analyses OSS licences with the aim of highlighting two main aspects. First, we investigate which licences better guarantee the contributor or the user of open source software to develop 'proprietary' software together with open source software and commercialise them as a whole work or separately. Second, we analyse whether and to what extent the licences guarantee that open source software cannot be appropriated by *free riders* (i.e., individuals or firms that develop proprietary software by using open source software without complying with the OSS rules). These specific properties, together with the general ones analysed in Table 4.1, make it possible to distinguish the OSS licences that can be used to support sustainable business models from OSS licences that do not guarantee software developers and distributors to extract rents from their activities.

We analyse four OSS licences models, adding to the general characteristics illustrated in Table 4.1 the following properties that are specific to OSS software:

- *'Mixability' with other OSS licensed products.* We analyse whether the software under a specific licence can be combined with and distributed as a part of other OSS-licensed products;
- *'Mixability' with proprietary licensed products*. We analyse whether the software under a licence can be combined with and distributed as a part of proprietary-licensed products;
- *Degree of openness protection*. This amounts to see the degree of protection of the users' rights to use, modify and distribute software: we analyse whether the licence preserves from the risk that part of the covered code is appropriated by individuals or organisations who violate the rights of contributors and users. The maximum degree of protection is ensured by the 'copyleft' property. As mentioned before, 'copylefted' software is software distributed with the source code and whose distribution terms do not allow the distributor to add any restrictions that limit the right of using, modifying and distributing the software;
- *GPL compatibility*. A license is compatible with GPL when the code under its terms can be combined with modules of GPL-ed covered code to form a larger work that could be distributed.

The license schemes analysed here - GPL, LGPL, MPL/NPL and BSD - were selected because each represents a template for many other subsequent licences approved by the Open Source Initiative. In what follows, we briefly describe the history and peculiarities of these four templates and illustrate their main differences (see Table 4.2)[31].

4.3.1. The GPL

The GNU General Public License (GPL) was developed by Richard Stallman in 1988 to prevent that parts of GNU[32] Project's code could be subtracted from free distribution among users and developers. The first version of this licence was applied to the free text editor Emacs and then made extendable to any application in 1989[33]. The GPL has the highest degree of openness protection among the OSS/free software licenses. This licence assures that the source code available and free through a complex set of rules such as: a) the 'copyleft' property claiming that distributors cannot add any other rule on the covered code to constraint the users' freedom to copy, modify and distribute the source code (neither they can charge a license fee); b) the 'viral' clause requiring that any derivative work based on software covered by the GPL, as well as any new code merged with GPL-ed code, must also be distributed under the GPL conditions[34].

These two rules are intended to ensure that a software developed freely by a community of volunteers cannot be appropriated by producers and distributors of proprietary software. They also tend to discourage the use of GPL-ed code in commercial development by traditional software companies and other for-profit organisations.

The introduction of the GPL is viewed by its advocates as a major challenge to the established institutions of intellectual property right. By contrast, legal experts claim that it can be simply viewed as a new type of contract through which copyright holders commit themselves to a non conventional use of their patrimonial and moral rights over their intellectual works.

What is certainly true is that the diffusion of the GPL through software products like Linux has challenged traditional licenses and established business strategies.

For the purposes of commercial distribution, however, the GPL is problematic since it discourage the combination of proprietary software and open source software. The 'viral clause' is problematic for

---

[31] Appendix B contains a detailed description of the characteristics of these OSS licenses.
[32] The GNU Project was launched in 1984 to develop a complete free Unix-like operating system - the GNU system (GNU is a recursive acronym for ``GNU's Not Unix''). The terms "free" software refer to users' freedom to run, copy, distribute, study, change and improve the software (see www.gnu.org and www.gnu.org/philosophy/free-sw.html for further details).
[33] For a GPL detailed history see www.free-soft-org/gpl_history.
[34] A derivative work of a GLP-ed program is defined as "any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof".

those who aim to use GPL-ed code to develop proprietary software and for proprietary software developers who aim to convert a proprietary software into an open source software. Consider, for example, a proprietary software which share some source code with other proprietary software of the same firm or includes third party proprietary code. Once the proprietary software becomes GPL covered software all other products that share with it a piece of source code become derivative works and as such must be comply with the GPL (see Hecker, 2000, p. 12). For a detailed description of the GPL see the Appendix B.

4.3.2. The LGPL

The GNU Library General Public License (LGPL) was developed by the Free Software Foundation in 1991 in order to permit the use of free libraries of programs into 'proprietary' executable programmes. The LGPL share with the GPL all restrictions described before. Unlike the GPL, however, it allows a proprietary programme to call LGPL–ed code (e.g., routines) without the derivative work becoming a GPL work. This makes the LGPL less restrictive than the GPL allowing the use of the code in proprietary works. Only changes to the LGPL library itself have to be distributed under the LGPL terms. The LGPL license is completely compatible with the GPL license.

4.3.3. MPL/NPL: The Mozilla Public Licence and Netscape Public Licence

The Netscape Public Licence (NPL) was created in 1998 by Netscape in order to distribute its browser (Navigator) under an open source licence. At the same time, they created the Mozilla[35] Organisation and the mozilla.org web site for stimulate developers to collaborate on this project. The Mozilla Public Licence (MPL) is virtually the same as NPL, except that NPL contains a set of rules that Netscape reserves for itself, i.e., NPL gives to Netscape the privilege of re-licensing modifications made by contributors under other distribution terms and to keep private any modification that they want to use for specific purposes. This provision is due to the fact that when Netscape decided to make its product free, it had already signed contracts with companies that committed Netscape to distribute the software Navigator under closed licensing terms. As a matter of fact, MPL was created in order to solve this problem.

4.3.4. The BSD licences

The BSD licence was originally used for the Unix distribution released by the University of California at Berkeley (BSD stands for Berkeley Software Distributon). BSD is a non-copyleft, free

[35] Mozilla is the name of both the open source project to create a free web browser and the browser itself.

software licence and is the most permissive among the OSS/Free Software licences. The copyright holder gives the permission to use, copy, modify and distribute the software almost without limitations, provided that a copyright notice and a disclaimer of warranty is included in each copy distributed. Furthermore, the underwriter has to be fully informed of the rights provided by the licence. It is possible to modify and distribute the software in the form of object code without distributing the source code.

The comparison of OSS licences reported in Table 4.2 points out several commonalties but also some relevant differences among them.

The first part of the table, which includes the same characteristics included in Table 4.1, suggests that OSS licences share most basic characteristics, except for a few cases, i.e. the BSD licence does not include initial developers special rights. Moreover, the conditions under which the distribution is allowed with the BSD are slightly different than with the GPL since the BSD does not include any obligation to distribute the source code nor limitations on the possibility to sell the software. In addition, the MPL includes explicit rules in order to solve third party claims and litigations, conflicts and violations. It is also more compatible with national laws both because it does not contain any 'viral' clause and because it explicitly claims that the license shall be subject to the jurisdiction in which the software is distributed.

The analysis of characteristics specific to OSS licenses suggests that licensing models like the MPL and the BSD seem to be more appropriate for business actors compared with the GPL licence, which does not allow to merge source code under GPL with proprietary software. As mentioned before, the 'viral clause' create problems for individuals and business firms that aim to develop or distribute software programs under different licence schemes. The LGPL license presents almost the same problems as the GPL but they can be used in commercial proprietary software that uses LGPL libraries without modifying or including them in the executable code.

By contrast, the MPL and the BSD licences do not impose any 'contamination' clause on OSS users and as such are compatible with proprietary software. Compared to the MPL, the BSD license does not impose restrictions on source code availability but, on the other hand, it has the lowest degree of protection of openness so that the code distributed can easily be appropriated by developers or distributors of proprietary software.

Finally, Table 4.2 shows that all OSS licenses models allow for distribution fees (not license fees) and therefore provide private incentives to commercial distributors.

The Appendix D reports data on the diffusion of different licenses in open source software projects. These data were obtained by analysing over 40,000 open source projects at different development stages (SourceForge, 2002). Over 94.9% of all projects adopt an OSI approved license and about 80% a GNU/GPL or a LGPL license.

The analysis of these data suggests that despite the existence of 32 OSI approved licenses (most of them created to better perform in the business world), the great majority of open source software is distributed under GPL terms. In our view this could depend on several factors.

First, the GPL license strongly prevent any attempt to include in commercial 'proprietary' products any software developed by individual programmers or by new, small business companies that develop and distribute OSS. Second, there could be a 'lock-in' effect due to the fact that most OSS actors are either developing software based on GPL-ed software (Linux, for example) or are using GPL-ed software to develop their own applications. Finally, it is possible that free software 'evangelists' use GPL as an instruments against commercial software and push for GPL-ed software diffusion. Although these hypothesis have not been tested yet, our interviews strongly suggest that commercial players operating in the OSS world perceive the GPL as the licensing scheme that better protect their development efforts from free riding behaviour while permitting to benefit of a large pool of talented developers. Also, the GPL licensing scheme ensures new entries that the OSS they develop is not appropriated by their own customers.

Table 4.2 OSS licences

LICENCE	GPL	LGPL	MPL / NPL	BSD licence type
Product Examples	Gnu-Linux, Red Hat 7.2, SuSe	AA-lib, PHPLib, Quicktime for Linux	Netscape Navigator	Xfree86, Tcl, Wine
Source Code Availability	Yes	Yes	Yes	Yes
Right to Use	Yes	Yes	Yes	Yes
Right to Copy	Yes	Yes	Yes	Yes
Right to Modify	Yes	Yes	Yes	Yes
Right to Distribute	Yes, conditioned to a) source code availability; b) notice about modifications	Yes, conditioned to a) source code availability; b) notice about modifications	Yes, conditioned to a) source code availability; b) notice about modifications	Yes, conditioned to attach copyrights notice and disclaimer of warranty and limitation on liability
Moral Rights Protection	Yes	Yes	Yes	Yes
Initial Developer 'special' rights	Yes	Yes	Yes.	No protection.
Is the software under this licence *gratis*?	Yes, it is generally *gratis*, but fee is allowed and sales are not banned	Yes generally *gratis*, but fee is allowed and sales are not banned	Yes generally, but both sale and a fee are allowed	Yes generally *gratis*, but there are not rules that ban sales and fees
Warranty / Liability / Claims	Total disclaimer of warranty; limitations on liability	Total disclaimer of warranty; limitations on liability	Total disclaimer of warranty; limitations on liability	Total disclaimer of warranty; limitation on liability
Third Party Claims	No explicit rules	No explicit rules	Yes, explicit rules are included	No explicit rules
National Laws	Problems of compatibility could arise	Problems of compatibility could arise	High compatible	High compatible
Litigations / Conflicts / Violations	No explicit rules	No explicit rules	Yes, explicit rules	No explicit rules
SPECIFIC CHARACTERISTICS OF OSS LICENCES				
'Mixability' with OSS Licensed Product	Yes	Yes	Yes	Yes
'Mixability' with Proprietary Licensed Product	No	Yes	Yes	Yes
Degree of 'openness' protection	Maximum protection by 'copyleft' property	Maximum protection by 'copyleft' property	High-Medium Protection	Medium-Low Protection
Is the licence GPL compatible?	-	Yes, high compatibility	Partial compatibility	Partial compatibility

4.4 Other OSS licenses

In addition to the licenses described above there are a considerable number of other licenses allowing source code modifications and distribution. Among them, the licenses approved by the OSI (Open Source Initiative) are 32. The OSI list include GPL, LGPL, MPL and BSD, but not the NPL license that gives Netscape proprietary rights for modifications made to the software. Netscape contends that the last version of its license, the NPL 1.1, satisfies the OSI criterion but the license has not been included in the list yet. Most OSI approved licenses can be clustered in four main groups. As Table 4.3 clearly shows, the majority of these licences are modelled around the MPL and the BSD templates.

Table 4.3 OSI Approved Licenses grouped by similar characteristics

MPL like licenses	BSD like licenses	Others
▪ Sun Public license ▪ Motosoto Open Source License 0.9.1 ▪ Jabber Open Source License ▪ Nokia Open Source License ▪ Ricoh Source Code Public license ▪ Sun Industry Standard Source License (SISSL)	**Pure BSD** ▪ MIT License ▪ University of Illinois/NCSA Open Source License ▪ X.Net License **BSD plus additional rules** ▪ Apache Software License ▪ Eiffel Forum License v.1 ▪ Intel Open Source License for CDSA/CSSM ▪ Sleepycat License ▪ Vovida Software License 1.0 ▪ W3C License ▪ Zope Public License	▪ Zlib Licence ▪ Open Group Test Suite License ▪ IBM Public License 1.0 ▪ Common Public License 0.5 ▪ Apple Public Source License ▪ MITRE Collaborative Virtual Workspace License (CVW License) ▪ Nethack General Public License ▪ Artistic License ▪ Q Public License ▪ Python License ▪ Python Software Foundation License

4.4.1.MPL like licenses

- The **Sun Public License** is obtained from MPL 1.1 by substituting the term "Mozilla" with "Sun".
- The **Motosoto Open Source Licence 0.9.1** and the **Jabber Open Source License** are derived from MPL applying some 'cosmetics' changes. Furthermore, both licenses explicitly claim that 1) the license grants exclusively the right indicated in it, and does not apply to any other intellectual property right hold by the licensor; 2) the license does not grant rights on licensor's trademark, and 3) it does not prevent licensor to license covered code under different terms.

- The **Nokia Open Source License** (NOKOS Licence) is derived from the MPL 1.1. Nokia made the following changes to the MPL[36]: 1) Nokia will be the only user of its license, that is in addition to using the company name throughout the license, it removed the provision of the MPL that permits others to use it as a template for new licenses; 2) Nokia limits its liability to US$50; 3) This license is to be interpreted under the copyright law.
- The **Ricoh Source Code Public Licence** is a mix between MPL v. 1.0 and MPL v. 1.1 plus explicit rules on trademark usage. For example rules concerning contract termination are drawn from MPL 1.1, whilst both rules about grants given up by the copyrights holder and contributor, and rules on claims, are from imported MPL 1.0. The trademark usage clause claims that: 1) All advertising materials mentioning features or use of the Governed Code must display the acknowledgement that the product includes software developed by Ricoh; 2) The names "Ricoh," "Ricoh Silicon Valley," and "RSV" must not be used to endorse or promote contributor versions or larger works including the original covered code without the prior written permission of the company; 3) Contributor versions and larger works including original covered code cannot be called "Ricoh" nor may the word "Ricoh" appears in their names without the prior written permission of the company.
- The **Sun Industry Standard Source Licence (SISSL)** is a MPL like license but distributions obligations are different: 1) only the Initial Developer can grant, *with* the license agreement, any right to copy, modify, and distribute to users. It is worth recalling that the MPL license instead contains two license models, one from the Initial Developer and another one for other contributors, and sets the rules about third party claims and patents that contributors have to follow; 2) any modification has to comply with a standard defined in the license body; otherwise, 3) like in MPL, any deviation from the standard protocol and the documentation of the process of modification have to be published. Modifications have to be distributed in form of source code.

4.4.2. The BSD like licenses

It is possible to classify the BSD like licenses in two groups: 1) licenses created by university institutions that simply adopt the scheme of BSD *(pure BSD)*; 2) BSD like licenses that add further provisions such as trademark and distribution clauses *(BSD plus additional rules)*. Among the latter, the licences enforced by companies or organisation that commercialise their products, include a clause on trademark (Apache, Vovida, and Zope).

[36] These differences are extracted from OSI commentary at www.opensource.org/licences.

- The **MIT License** is a simplified version of the BSD license.
- The **University of Illinois/NCSA Open Source License** is obtained directly from BSD licence.
- The **X.Net License** is a simplified version of BSD license.
- The **Apache Software Licence** is a BSD license plus a trademarks options and an acknowledgement requirement.
- The **Eiffel Forum Licence v. 1**, is a BSD license plus a distribution rule. The license allows the object code distribution, but if the binary programs depend on a modified version of the covered code, the distributors must publicly release the modified version of the source code.
- The **Intel Open Source Licence for CDSA/CSSM** is obtained by adding to a BSD licence an export notice that attributes to the licensee the responsibility to comply with any export regulations applicable in the licensee jurisdiction.
- The **Sleepycat License** is a BSD like license plus a distribution provision ensuring the availability of complete source code that resembles the GPL rule.
- The **Vovida Software Licence 1.0** is a BSD license plus a trademark clause stating that the software derived from Vovida's "Vocal" software, cannot be called "Vocal" nor it can contain the term "Vocal" in their name without prior written permission by the company.
- The **W3C License** is a BSD like license plus a distribution rule. The license gives the right to use, copy, modify, and distribute the original covered code with or without modifications. Furthermore, it states that the distributor has to attach to all copies of the software, including the modifications, any pre-existing intellectual property disclaimer, notice, or term and condition. Notice of any change or modification to the W3C files, including the data of changes, are required.
- The **Zope Public Licence (ZPL)** is a BSD like licence plus a distribution and a trademark rule. The license states that if a file is modified, the 'underwriter' has to include in the following distribution a notice stating what has been changed, when the modification occurred and the author of modifications. Furthermore, ZPL states that the licence does not give to the user the right to use Zope Trademarks or Servicemarks. The use of them is regulated in a separate agreement.

The other OSI approved licenses, amongst those listed above, show peculiar characteristics mainly associated to the type of product, market, organisation or company that distribute the software.

Appendix B reports the complete list of licenses approved by the Free Software Foundation. The list includes three categories of licenses: i) GPL-compatible free software licenses; ii) GPL-incompatible free software licenses; and iii) non-free software licenses (http://www.gnu.org/licenses/licenses.html). All free software licenses - both in the first and in the second category - guarantee the distribution of source code and the freedom for users to copy, modify and distribute it. They are approved by FSF, but they differ with respect to the GPL-compatibility. The difference between the first two categories depends on whether the code under the license can, or cannot, be combined with modules of GPL-ed covered code to form a larger work that could be distributed. GPL-incompatible free software licenses are those that cannot be combined with modules of GPL-ed in the sense stated above, whilst GPL-compatible licenses can. Non-free software licenses are those for which the right to deal with source code and its distribution are limited in some way, therefore FSF does not approve them (see below the reasons to reject Apple Public License). Comparing free FSF approved licenses with OSI approved license we note that the FSF has approved 15 license more than the OSI. We have not analysed yet the reasons for this differences. There are five licenses (Guile License, Gnu ADA Compiler License, Standard ML License, iMatrix License and OpenLDap License version 2.7) that are GPL compatible but are excluded from the list of the OSI. Public licenses adopted by Arphic, xinetd, Open Ldap 2.3, Vim, Phorum 1.2, Latex, Netizen, Interbase 1.0, Freetype, and PHP are free software-incompatible licenses 'approved' by the FSF but not are OSI approved. Finally, note that FSF does not consider the Apple Public Source License (APSL) as a free software license, whilst OSI has approved it. This is because Apple Public License claims that any modified version developed by an organisation, for internal use or for distribution within the organisation (except for R&D use and/or personal use) has to be published. APSL then seems to violate users' and developers' freedom. The reasons why OSI does not recognises this violation will be subject of further analysis.

5. The economics and business of open source software

Several recent works have studied the Open Source Software phenomenon focusing on different issues: the incentives to develop software without monetary compensation, the heterogeneity of users preferences that induce their involvement in the innovation process, the competition between OSS and proprietary software, the business models for making profits from open source software (see, for example, Lerner and Tirole, 2000; Lakhani and von Hippel, 2000; Franke and von Hippel, 2002).

These studies point out different reasons for the rising diffusion of OSS products. Our work focuses on the incentives and organisation of different agents involved in the development of OSS –academic institutions, communities of hackers, commercial developers, pure distributors and, more recently, established proprietary software firms.

These agents contribute to technical progress and diffusion of OSS in various ways. They show different strategies – from vertical integration to specialisation in specific activities such as development of core products (e.g., kernels), packaging and commercial distribution of legacy OSS products (e.g., Linux and Sendmail).

Figure 5.1 illustrates different actors and their location in the space of R&D and commercialisation activities.

Figure 5.1. Actors and Activities

Activities / Institutions	*Basic / applied research*	*Development*	*Distribution*	*Services*
Academic institutions	XXX	X	-	-
Non profit developers (individuals and OSS communities)(1)	?	XXX	X	X
OSS commercial developers	?	XXX	XX	XX
OSS pure distributors	-	-	XXX	XXX
Proprietary firms (2)	XXX	XXX	XX	XX

Legend: X = of limited importance; XX = important; XXX = very important.
Notes: (1) This category includes sophisticated users such as web administrators and scientists engaged in developing tools or software programs to support their core research activity. (2) This category includes producers of complementary products like computers and books.

In the OSS world, besides the academic institutions (that distribute their free software) and hackers communities associated to the FSF and the OSI, who focus on development activities and are not engaged in distribution, there are new companies developing new software, with the contribution of voluntary developers, which aim to make profits not primarily from software (which can be given for

free or for a limited fee) but on related services. These firms download open source software from the Internet, package the software and sell installation and maintenance services. They face low barriers to entry, because they make very small investments in software development - e.g., add-ons like GUIs and customisation of kernels to specific user need.

At the opposite side, incumbent proprietary firms carry out different software development activities, including programming, and typically do not disclose the source code of their programs. They also provide technical support and related services. These companies do not have incentives to disclose information, but on the contrary to protect it by means of various instruments of legal protection (copyright, patents and trade secrets). For reasons discussed later, some of these firms have begun to disclose some source code. However, sometimes established proprietary companies have disclosed their source code only in later advanced stages of their innovation process, or after having internalised most of the returns from innovation. In some instances, the reaction of commercial firms to OSS appear to be too late and too little. For instance, Netscape's decision to make part of its browser source code (Mozilla) open is probably too late if we consider that in 1998 the fundamental features of Nescape's Internet browser have been already developed and probably only minor improvements remained to be introduced. Moreover, a commercial firm cannot easily credibly commit itself to an OSS project (Lerner and Tirole, 2000).

All individuals and institutions engaged in OSS production and commercialisation share a common key feature that is the disclosure of the source code to users. The reasons why source code is disclosed vary across different categories of agents. In general, however, what is most challenging to the economic theory is why rational agents voluntarily reveal their information if they know that its economic value will tend to zero afterwards. If we believe that OSS can be represented as a 'gift economy', where non-pecuniary motives, such as the desire to reciprocate or the desire to avoid social disapproval, are more important than monetary incentives, then the source code is used to signal to the OSS community our adherence to the social norms of that community.

Then the question arises as to what are the benefits and costs of being part of that community. A related question is whether this community can be compared with the community of 'open science'. As David (1991) and Dasgupta and David (1987, 1994) have noted, individuals and organisations engaged in open science share social norms and institutions that are different from those of industrial research or 'technology'. Scientists aim at full disclosing their discoveries, public replication of results and verification of claims. They are motivated by non monetary benefits such as priority in discovery and scientific reputation established through a collective, peer review system. Their professional (monetary and non monetary) awards depend on reputation.

But not all participants engaged in OSS production and distribution seem to be motivated by the typical incentives of the 'gift economy'. Commercial developers and distributors, like proprietary software firms that have begun to disclose their source code, are most probably motivated by the pecuniary incentives of the 'exchange economy'. They share with the traditional model of software production norms and incentives to engage in innovation that are typical of the 'technology' community – i.e., expected profits and 'exclusion' mechanisms or appropriability systems (lead time, trade secrets, copyright and patents). In the case of software, more than in other technologies, 'exclusion' mechanisms, including patents, impose limited disclosure obligations over proprietary information. As discussed before, even the patent law does not impose the disclosure of the source code to inventors. Why then profit oriented institutions such as computer manufacturers and software publishers engage in the production of OSS and therefore voluntarily disclose their proprietary information? As we discuss later on, a main reason for this strategy is the need to react to the threats arising from the commercial success of OSS products like Linux and Apache. Another reason is due to the fact that the distinction between basic research or open science and applied research or technology is arbitrary and blurred. Profit oriented institutions are then spurred to carry out basic research with the consequence that they have also to accept to some extent the norms of 'open science'.

In the subsequent sections we discuss first the motivations of OSS developers, the similarities and the differences between OSS and open science. Secondly, we analyse the competition between OSS and proprietary software. Finally we briefly mention some potential implications of OSS for growth and social welfare.

5.1 Incentives to engage in open software: 'gift culture' or 'exchange economy'?

As mentioned before, OSS developers are driven by different motivations. The economic literature has tried to understand these motivations by focusing primarily on the behaviour of hackers who participate in associations like the FSF and the OSI. The literature has highlighted the following main motivations:

- ethical and political reasons;
- altruism;
- reputation inside the community;
- need to adapt software available on the Internet to their own requirements and to solve specific problems;
- pure fun.

Ethical and ideological reasons motivate a large portion of the hacker community, that Raymond (2000) defined as the "very zealous and very anti-commercial" segment of the OSS world. The Free Software Foundation developed a great amount of *free* software to guarantee the maximum diffusion of knowledge and preserve users' right to freely copy, modify, and distribute the source code. Users' rights are opposed to the IPR. As Richard Stallman, the founding father of the FSF, argued "copyright is not a natural right, but an artificial government-imposed monopoly that limits the user's natural right to copy" (Stallman, 1999).

But why inventors should give up their property rights? One reason, according to the advocates of the OSS, is that OSS contributors do it for the "joy of programming", that is the satisfaction derived from writing a good software and to solve a problem, and "personal itch" (Raymond, 2001). Another motivation is "altruism". Eric Raymond, one of the founder of the OSI and developer of the OSS *Fetchmail*, in his "Homesteading the Noonsphere" (2000) argues that in a world characterised by abundance of resources, a 'gift economy' tends to substitute the 'exchange economy'. Moreover, individuals' reputation rises with the gifts they offer to the community and not with the goods they can exchange. In a software community, which is usually characterised by abundance of resources (resources, as Raymond wrote, are disk space, network bandwidth, computer power), the only available measure of success is reputation among peers. Then the community members are motivated to contribute freely, and with high-value gifts, to the software development.

Lerner and Tirole (2000) agree that reputation is one of the driving force of this community but they disagree with the 'altruistic' hypothesis submitted by Raymond. They assume that developers are rational economic agents who decide to allocate their efforts to develop OSS rather than proprietary software on the basis of a cost-benefit analysis. Lerner and Tirole distinguish between direct and delayed benefits. Direct benefits derive from solving specific problems - i.e. fixing a bug - faced by the programmer or somebody else or taking immediate monetary compensation (whenever the developer works in a company that develops OSS) while 'delayed benefits' arise from signalling to the market to be a good programmer. These benefits can take the form of a good or a better job position, access to venture capital and an improved reputation amongst experts. Then ego satisfaction arising from peer recognition can be transformed in 'economic' compensation. Then, according to these authors, developers are mainly driven by career concerns.

Obviously, these benefits have to be compared with opportunity costs. Lerner and Tirole note that in the OSS community opportunity costs are lower compared with the proprietary software market because of the "alumni effect": "because the source code is freely available to all, it can be used in

schools and universities for learning purposes; so it is already familiar to programmers. This reduce their cost of programming for UNIX, for example" (Lerner and Tirole, 2000).

Dalle and Jullien (2001a) have also underlined the role of career concerns that are becoming more and more important with the emergence of service companies (that they call "ancillary business companies") offering support, maintenance and customisation for installed open source software. However, they underline that this motivation can explain only the kernel developers' behaviour. Services companies are often created by kernel developers that can obtain venture capitals thanks to their prior signalling activity described before; at the same time these companies employ kernel developers, so that open source contributors have incentives to signal their skills by sharing their code with the community. These motivations do not apply to "obscure developers" who develop less creative parts of code – because for them expected profits associated to their contribution is low.

Among "obscure developers" there are most probably "innovative users" (Von Hippel, 1988), who develop software for their own purposes. Sophisticated users may contribute to software development or debugging because they do so for solving their own needs, or just for fun (von Hippel, 2001).

Their opportunity costs are probably low, especially when the costs of outsourcing the same development tasks is made high by idiosyncratic, sticky information. Von Hippel (2001) observes that if individual users have sufficient incentives to innovate, to reveal their knowledge freely and possess the capability to do so, then a community of innovation and trial arises. When the product developed by these innovative users can compete with commercial production and distribution, as for open source software, it is possible that a community of innovative users can compete with traditional, supply-led innovation. Like any other economic agent, users participate in the development and innovation process if they expect the benefits of innovating will exceed their costs. The costs of participating in development activities can be very low, as Lakhani and Von Hippel (2000) show in the case of the Apache Usenet community which offers "field support" to users of the Apache web server software. They find that the action of delivering help is really costless. Information providers spend just few minutes to reply if they know the answer, or they can look for it on the Apache forum, eventually learning more on the product (in this case they extract an immediate benefit). Lakhany and Von Hippel (2000) also find that developers often disclose their knowledge if they think that other have the same information and are going to disclose it; and in this case the loss of intellectual property value associated with disclosure is zero. Since priority is important in this community, imperfect information and expectations about what knowledge other contributors could offer to the community should drive to a *disclosure race* whose long term consequences for the economics of OSS and proprietary software are not fully explored in the literature.

Moreover, Lakhani and von Hippel point to the organisational limits of this model. They note that in the Newsgroup discussion forums on the Apache Usenet a relatively limited number of questions are posted to the system if compared with the very large number of web sites run by Apache server (about 8 million sites in the early 2000). They also estimated that only about 100 information providers account for over 50% of response messages. They conclude that if the number of questions posted to the Usenet rose significantly the system would be put under a tremendous pressure. This system has proved to be robust to a rising, albeit limited number of questions, thanks to other complementary support services such as specialised books, on-line journals and FAQ databases. But the growth of a support service industry could be critical to the future growth of this and similar systems (Lakhany and Von Hippel, 2000).

A recent empirical survey performed by Hars and Ou (2001) provides further support to the hypothesis of economic rewards - that is the opportunity of direct or indirect monetary compensation, self marketing and peer recognition. Among these, the desire of improving their own skill base (with 88.3% of responses) is the most relevant for all interviewees. The survey also shows that open source communities are composed of different actors, including students, hobbyists, programmer paid for developing open source software, and different incentives structures seem to apply to different groups of developers. For example, hobbyists and students are the ones most motivated by altruism and sense of community, while programmers paid for OSS development are those most motivated by selling the product and self marketing.

Finally, pure fun leads many people to develop software and fix bugs. Especially in the segment of software games, students and in general young people, greatly contribute to the product improvement (Jeppesen, 2001).

All motivations discussed above mainly apply to individual developers, whilst the motivations of companies, i.e. software companies providing OSS and services and/or large commercial software companies, for developing OSS are discussed later. An interesting issue for future research is whether OSS represent a stage in the individual developer's early training stage which is followed by a different pattern of development in subsequent stages of her career.

5.2 Open software and open science

The open source developers motivations illustrated before do not clarify completely a key questions raised before: is open source software comparable to open science? If so, does it matter that a scientific discover in software be produced by a OSS institution or proprietary software one? The links of OSS with open science have been highlighted by the economic literature (Lerner and Tirole, 2000).

According to a fascinating theory, the birth of open science can be traced back to the sixteenth and seventeenth centuries as the result of the competition among imperfectly informed noble patrons concerned with the "ornamental" and "status-enhancing" utility of the sponsorship of natural philosophers (David, 1991). The formal sophistication of methods used and outputs yielded by natural philosophers made it extremely difficult to the patrons to observe the true ability of scientists ("qualifications and comparative merits") and gave rise to a typical principal-agent problem to which the institution of open science provided an efficient solution. Imperfectly informed patrons could delegate the evaluation and selection of experts to other experts drawing on a set of norms and organisations (full disclosure, reputation and priority) that modern science has inherited from the feudal system with minor changes. Over time, with the formation of national states and universities, open science became largely independent from patronage. Today, due to its public good nature, two-thirds of basic research is founded by the state (Rosenberg, 1990).

Whatever the origins of OSS, the incentive structure of the OSS community (especially hackers and non profit organisations) is similar to that of today 'open science'.

Like scientists, OSS developers share a virtual context and a common language that favour the exchange of information. While refereed journals and public conferences represent the natural loci for disseminating research outputs, the Internet and the projects leaders control the flow of information amongst developers and certify the quality of each contribution. After this development stage, the market (or more generally the community of users) will test the innovation. Unlike the proprietary software organisation, not only the number of participants in the scrutiny of each innovation is potentially very large, but also the nature of evaluators is different. In proprietary software, the number of people that participate in a process is necessarily limited by the size of the internal network of researchers and by reasons of secrecy. Moreover, for proprietary software the ultimate scrutiny is made by the market.

There are also some problems that complicate the comparison with open science. First, the system of validation and replication of results in most branches of natural sciences, including computer science, is very rigorous and the access to the status of 'expert' requires long formal training and the compliance with national and international certification standards. OSS advocates argue that OSS represents an important training ground for computer skills. This is probably true but the training system and related standards are not specific to OSS, as they have been developed by the academic communities, often in collaboration with technical communities. Also, today several proprietary software firms (including Microsoft recently) have agreed to disclose their source code for experimental and training purposes.

Second, discoveries and theories in many scientific disciplines, including computer science, are published in technical and scientific journals and are open to peer evaluation[37]. These mechanisms work for the computer science in general, not for OSS in particular. It is possible, of course, that a proprietary software firm has the incentive to retard or give up the publication of a patented or copyrighted invention. But this is an issue for empirical analysis. As a matter of fact, some authors have pointed out that the growth of the patent system in the US, and the involvement of universities in the patent race, may have yielded distortion effects on the allocation of resources to inventive activity and the diffusion of knowledge. A careful analysis of patents (USPTO and EPO) and publications in technical journals (e.g., IEEE Computer Magazine, IEEE Transaction of Software Engineering, IEE Software Engineering Journal) could probably help to understand these effects.

Third, as noted by Rosenberg (1990), the separation of open science (and basic research) from technology (applied research) is "a highly artificial and arbitrary" one. This is demonstrated by the fact that many celebrated scientific discoveries have resulted as a by-product of applied research, i.e., problem solving activities in the wine industry (bacteriology), mechanical engineering (thermodynamics) or telecommunications (radio astronomy). This mixture of pure and applied science is a reason why profit oriented institutions engage in pure research or open science. In the computer and software industry too established firms like IBM and Microsoft invest their money in basic research to have access to new knowledge either directly (from their in-house research activities) and indirectly (by gaining access to outside discoveries). These firms know that in order to have access to the scientific community, they have to pay an 'entry ticket' which consists in resources dedicated to pure research and a minimum of disclosure of results that is needed to be accepted by the community. The question is then what does exactly mean to be part of the open science community and whether the OSS is different from proprietary software because it is closer to open science.

Fourth, the organisation of open science is typically different from that of the OSS communities. Experimental disciplines need expensive research facilities and formal organisation. As Kargon, Leslie and Schoenberger (1992) have pointed out, "Big Science is marked, as is well known, by the hierarchical organised team, led by a principal investigator-manager, with co-principal investigators…we can call this kind of organisation a *vertical integration* within the scientific enterprise" (p. 335). It is possible that the scientific enterprise in software is different but the experience of academic departments such as Stanford University or Carnegie Mellon University and the basic research conducted in industrial research labs such as IBM or Xerox Parc suggest that research teams can be large and well organised. This means that the idea of scientific research as a virtual college activity is largely undocumented in the

[37] As discussed before, some argue that in the case of software there is a lack of documented scientific knowledge which makes it difficult to establish priority (Cohen and Lemley, 2001). This is probably true for minor, incremental innovations.

empirical evidence. Of course, there are many networks of researchers which cut across regional and national boundaries. However, the physical dimension of research is still an important characteristic in the organisation of scientific activity (this is also demonstrated by the regional concentration of R&D in clusters around universities or large industrial research or state-owned labs). In this respect, the OSS model represents an alternative to the current dominant model of scientific research organisation. And it is to far to be clear how it will behave in condition of rising size and complexity of research targets. Also, it is not clear whether the scale and low concentration of resources represents a constraint in the pursuit of radical technical innovations and scientific discoveries which, almost by definition, are characterised by high costs and uncertainty.

5.3 The competition between open source and proprietary software

Before turning our attention to the business models emerged with the OSS, we analyse the competition between open source software and proprietary software. Can these two models coexist or will one dominate over the other? And what are the implications of the competition between the two models for economic growth?

A couple of recent theoretical papers address these questions. Dalle and Jullien (2001b) study the competition between OSS and proprietary software in a diffusion model which takes into account users heterogeneity (users have idiosyncratic preferences over the two models that in this context are treated as two alternative technological standards), different distribution of individual preferences (adoption thresholds) and network externalities. The simulation run by the authors show that the possibility that a new OSS standard replace a dominant proprietary technology depends on a series of factors:

i) the importance of local externalities (which are positively related to the number of agents in the neighbourhood which adopt the same technology) and global externalities (related to overall improvements in quality and performance during the diffusion of the technology);

ii) the initial distribution of potential adopters with absolute (idiosyncratic) preference for the new OSS technology. For instance, a initial mass of skilled users with a psychological and ideological motivations and a component of 'evangelism' helped the take off of Linux among servers while for the same reason Linux has never took off in the clients markets;

iii) the compatibility with the proprietary standard. The greater is compatibility the larger the opportunity to steal business to the incumbent standard because compatibility reduces switching costs;

iv) product differentiation is also important when the are two distinct sub-populations of potential adopters with heterogeneous preferences. With strong network externalities, product

---

But probably for major discoveries this is not the case. Again, this issue should be explored by empirical analysis.

differentiation helps to target market niches with absolute preference for OSS products and to gain rapidly market share.

Dalle and Jullien (2001b) also analyse the reaction strategies of proprietary software incumbents. The latter have two main choices: do nothing or modify their strategy as soon as the OSS competing standard reaches a critical market share. The potential reactions strategies are centred respectively on price reduction, R&D increase (with the aim of improving the quality and efficiency of the proprietary product) and distribution. The latter strategy in reality has been tried by incumbent proprietary firms, which have shifted from product selling to product renting. By renting a software programme against the payment of a royalty, customers gain immediate access to bug corrections and improvements while with the acquisition of a package product improvements come only with new releases of the product. The simulations show that no reaction strategy among those examined yield effects significantly different from the 'no reaction' strategy. As noted by the authors, these apparently surprising results indicate the strength of path dependence effects. Once a critical mass of adopters of the OSS standard has been reached reaction strategies become ineffective because the adoption choices are bounded by the network externality effects.

However, incumbents can try another more effective reaction strategy that is hybridisation. By hybridisation an established technology (e.g., combustion engine in automobiles) took a specific characteristic from a competing new technology (e.g., the electric starter from electrical engines). Hybridisation yields two potential results: first, a positive effect on adoption due to the improvements of the product; second, the strategy can modify the preferences of potential adopters – some adopters with absolute preference for the new technology may decide to adopt the old, improved technology. Simulation results presented by these authors show that this can be an effective strategy. As a matter of fact this strategy has been adopted by several proprietary software firms such as Netscape (Mozilla), Sun (OpenOffice) and IBM.

Saint Paul (2001) analyses the competition between OSS and proprietary software in a model of endogenous growth. He highlights two kinds of externalities arising from the OSS that is characterised by gift exchange and community values shared by philanthropic innovators. First, the typical positive externalities which reduce the costs of proprietary software. In line with the new growth theory, these externalities ("manna from heaven") are positively associated with the number of goods (competitors doing R&D activities) in the market (Grossman and Helpman, 1991). Second, philanthropic innovators produce also negative externalities due to the low (around zero) prices of OSS. The low price reduces the demand for proprietary software and as a consequence profits of proprietary software shrink. Therefore, OSS "steal business" from proprietary software. These negative externalities reduce

the ex ante incentive to innovation of proprietary software firms and have negative consequences for growth in the long run (even if the "manna from heaven" may have positive short term effects).

These consequences are even more marked if one takes into account in the model also the fact that philanthropic developers may have no great incentives to invent "plain new (free) goods rather than free versions of an existing patented good" (Saint-Paul, 2001, p. 13). Put in another way, the OSS sector benefits from positive externalities generated by the private software sector which further reduce the ex ante incentive to innovate by proprietary software inventors. The net effects on growth and welfare depend on the relative importance of these two forms of externalities. We must also consider that the producers of OSS generate another kind of externality in favour of the proprietary software. Non only the latter enjoy the cost externalities discussed before. They have also free access to OSS products that can be imitated and transformed into proprietary ones. This can be positive for growth but reduces the ex ante motivations of philanthropic innovators. The elaboration of strict license schemes (e.g., the GPL) by the OSS communities is intended to reduce the probability of free riding by proprietary software.

To summarise, the competition between OSS and proprietary software can have positive effects since it may stimulate innovation in a sector characterised by network externalities that reduce the room for competition among alternative standards. However, different types of externalities generated by the two sectors can have negative effects on the inventive rate and growth in the long run.

5.4 Business models

As discussed earlier, the boundaries of the OSS extend well beyond the communities of voluntary developers to include a variety of commercial companies which specialise in different activities – from distribution of open software and provision of related technical support services to development of open source software and provision of kernel customisation and technical support services. The market for commercial OSS is also populated by established proprietary software companies and hardware manufacturers.

In parallel with the diffusion of several new licence models associated with the OSS a variety of new business models have been introduced into the market. The new OSS-based business models compete with traditional business models based on proprietary software.

The fundamental ingredients of a business model in the software industry overall can be summarised as follows:

- The core business of the firm - new software development, improvement of existing products, software distribution, services;
- The organisation of the development process and the division of labour with other organisations– scale of the OSS developers team, type of software code disclosed and distributed, contractual agreements with suppliers and distributors;
- The sources of firm's revenues and related issues – software product pricing strategy, services billing strategy or and pricing of complementary products;
- The IPR strategy and the licensing model (open source vs. proprietary).

The traditional business model is characterised by internal software development (or subcontracted to third parties). R&D investments represent a fixed cost that is recovered usually through proprietary licence fees. The licence provides the right to use the software, but not to copy, modify and redistribute it. The source code usually is not disclosed with the object code and a price is charged for the acquisition of a particular copy of the programme (license fee) or for the temporary use (renting) of the programme. A separate price is normally charged for support services. Prices in this sector are not driven by marginal costs. Price discrimination is often operated across different users (for example business or academic customers) for the same product or service, according to the price elasticity of demand and the market power of customers relative to vendors. Finally, distribution and post-sales services are provided directly by the software company or by third parties – either affiliated vendors, exclusive agents or independent resellers. The value added by third parties varies with the type of

software distributed (its unit value, and the complexity of its installation and maintenance). In typical shrink-wrapped products, like PC operating systems and office automation applications, most of the value is produced by the editor (e.g., Microsoft, Symantec or Adobe).

As for OSS business models, there are various types of business models that differ mainly for the source of revenues (software development or services and distribution) and the licensing models.

A key characteristic common to most OSS business models is that the source code is open and several voluntary people share responsibility for developing, distributing and maintaining programs. This implies that the value delivered to customers results from the cooperation among several developers and distributors. In order to preserve the right to free access to the source code and avoid free riding that would undermine the motivations of voluntary contributors, this class of business models requires that specific licensing models are devised.

The pricing strategy is usually based on low prices for the executable code (including the cost of assembling, writing documentation, and development of complementary software such as graphical user interfaces development and hardware drivers). Higher prices are charged for the provision of installation, maintenance, training, customisation and other customer services. The bulk of the revenues then normally arise from services rather than software. For this reason it is important to focus on the potential market for this particular type of services.

Unfortunately, there are not significant data on open source-based software services. However, we can get an approximate idea of the potential market for these services if we look at the software support market overall.

Recent IDC (IDC, 2002c) estimates show that the worldwide software support market generated revenues of about $21 billions in 2001 (about 5.2% of the world market for IT services), with an 11% increase over 2000's $19 billion in revenue. IDC projected that the compound annual growth rate (CAGR) will be 13% from 2001 to 2006 and the total revenues are then expected to reach $38.4 billions in 2006 (Table 5.1).

IDC estimates that over the same time period support services for development tools will increase at a faster rate than those for applications and system software (16.3% vs. respectively 11.3% for applications and 12.1% for system software). Moreover, telephone support and software maintenance will maintain the largest market share but maintenance and remote diagnostics will be the best performers in terms of growth rate (35% and 55% CAGR respectively). According to IDC "these fast growing services reflect the importance of support technologies that enable proactive, self-service support models; it also highlights the movement in the support industry to transform the support organization from a reactive cost centre to a valuable, important, and revenue-generating business

model" (IDC, 2002c). Linux and other open source software environments account for less than 0.5% of total software support market in 2002 and, despite the volume of sales are expected to grow very fast between 2000 and 2001 (over 57% CAGR), its market share is expected to remain small (about 1.4% by 2006) (IDC, 2002c, p. 25).

Table 5.1 Worldwide Software Support Service Revenue by Product Type, Region, Operating Environment, and Service Activity, 2000-2006 ($M)

	2000	2001	2002	2003	2004	2005	2006	2001-2006 CAGR (%)
Product Type								
Applications	7,399	8,217	9,205	10,296	11,445	12,752	14,061	11.3
Application development	4,464	4,936	5,642	6,522	7,672	9,032	10,498	16.3
System Software	7,035	7,819	8,773	9,717	10,981	12,413	13,814	12.1
Total	*18,898*	*20,972*	*23,621*	*26,536*	*30,098*	*34,197*	*38,373*	*12.8*
Region								
Americas	10,665	11,714	13,027	14,663	16,670	18,934	21,261	13.0
EMEA	6,339	7,319	8,408	9,413	10,627	12,036	13,116	12.4
Asia/Pacific	1,894	1,939	2,186	2,460	2,801	3,227	3,636	13.4
Total	*18,898*	*20,972*	*23,621*	*26,536*	*30,098*	*34,197*	*38,373*	*12.8*
Operating environment								
Mainframe	2,842	2,773	2,674	2,621	2,572	2,567	2,571	-1.2
OS/400	621	619	612	605	598	587	556	-2.1
Unix	5,721	6,455	7,268	8,024	8,825	9,543	9,977	9.1
Linux/other open source	28	56	105	177	285	442	537	57.2
Other host/server	868	833	802	756	720	681	637	-5.2
Windows 32 and 64	7,254	8,628	10,450	12,542	15,205	18,336	21,949	20.5
JVM/platform independent	46	77	111	178	260	339	357	36.0
Mobile and embedded	378	492	594	707	835	988	1,151	18.5
Other single user	1,141	1,079	1,003	926	800	715	637	-10.0
Total	*18,898*	*20,972*	*23,621*	*26,536*	*30,098*	*34,197*	*38,373*	*12.8*
Service activities								
Telephone support	10,772	11,765	12,873	14,064	15,621	17,406	18,841	9.9
Remote diagnostics	661	839	1,087	1,433	1,926	2,599	3,530	33.3
Electronic support	1,890	2,307	2,834	3,450	4,214	5,130	6,140	21.6
Software maintenance	3,591	3,985	4,488	4,909	5,297	5,642	6,025	8.6
Onsite software support	1,814	1,846	1,984	2,149	2,348	2,565	2,801	8.7
Predictive/preventive maintenance	170	231	354	531	692	855	1,036	35.0
Total	*18,898*	*20,972*	*23,621*	*26,536*	*30,098*	*34,197*	*38,373*	*12.8*
Growth (%)	**NA**	**11.0**	**12.6**	**12.3**	**13.4**	**13.6**	**12.2**	

Source: IDC (2002c)

5.4.1. New business models centred on OSS

The Open Source Initiative (OSI) provides a list of companies actively involved in open source software production and distribution (Table 5.2).

Table 5.2 OSI List of vendors

IBM
Apple Computers
SGI
Cygnus Solution, Inc.
Cyclades, Inc
Linux Mall
Red Hat Software
Riverace Corporation
C2Net Software, Inc.
Netscape Communication, Inc.
Walnut Creek Software
Cobalt Microserver, Inc.
Whistle Communication, Inc.
Caldera, Inc.
Corel
ArsDigita
ActiveState
Lutris Technologies
Sleepycat Software, Inc.
Covalent Technologies

Source: **www.opensource.org/products.html**

As a first attempt to classify different business models, we can distinguish among three categories of business models:

1. business models centred on purely collective developed of software by a group of hackers (e.g., Linux or Apache). The source code is downloaded, assembled and sold together with support services by commercial distributors (i.e., RedHat, Caldera, Suse). The quality of the output is related to the individual and organisational capabilities of the group of developers (organisational features such as the number of project administrators and developers, the management style, and features related to the vitality of the project such as the number of fixed bugs, patches, external contributors etc.). The bulk of revenues come from services;
2. business models based on open source software initially developed by a company and then disclosed to the community of OSS developers (e.g. Zope). The quality of the software depends

both on the company initial R&D efforts and the community contributions. The revenues come mainly from software customisation and services.

3. mixed business models mostly adopted by commercial proprietary software developers. This category includes firms that make open the source code of products previously closed (e.g., Netscape-Mozilla). On many occasions, the firms that adopt this mixed approach develop both open and proprietary software. The revenues of this category of firms mostly derive from hardware sales and from complementary products sales.

Two critical aspects of OSS business models are represented by management strategy and marketing strategy.

- With respect to the management strategy, the key ingredients are represented by the quality of the core team dedicated to the management of the open source projects, the existence of an infrastructure for external developers, the modularisation of product design that allows the integration of incremental external contributions, including lead users.

- As far as the marketing strategies are concerned, although the final commercial product is provided with the source code, it is important that the product is provided in binary form which is easy to install for customers that are not interested in having the source code. A significant component of the marketing strategy is represented by the management of the legal procedures involving the licences of the pieces of software included in the final product and the licence accompanying the product must be carefully addressed. Another critical component is pricing. Some OSS business models recall a typical entry price strategy adopted to price (proprietary) information goods which requires low initial profits in exchange for a high stream of future profits. This strategy is quite popular in industries like cellular phones, elevators, photocopiers and printers. The acquisition of an initial large installed base of customers may set in motion a process of cumulative causation and increasing returns which is typical of industries characterised by network externalities (Murtha, 1998).

Drawing on the concepts described earlier and examples drawn from the www.opensource.org web site, Hecker (2000) has defined four specific types of business models that emerge in the OSS market.

The first model is called "Support Sellers". It is based on the provision of software without fee (or with a very small fee to cover the costs of distribution of the software) and the sales of services, support and documentation. Examples of this category of business model are Caldera, RedHat, Suse,

Mandrake, which package Linux kernel and utilities and provide support and documentation. Other firms like Cygnus Solution provide services for GNU compilers. The most common license adopted by these firms is the GNU/GPL. The revenues come from the sales of physical items (like media and hardware documentation) and services.

The "Loss Leader" model is characterised by the free provision of an open source software associated with a proprietary product which accounts for the bulk of their revenues. The typical licences used in this model are the Mozilla and BSD licences. Examples are Netscape and Sendmail.

The "Widget Frosting" business model is based on distribution of open source software together with a core business hardware product. The aim of the hardware company (for which software is a cost rather than a source of profits) is to get better drivers and interface tools cheaper. Examples are Corel / Netwinder and VA Linux.

Finally the "Accessorizing" model is based on the sales of accessories, like books or other hardware, with pre-installed open-source software. Examples of this business model are represented by O'Reilly Associates, SSC, and VA Research.

Other business models suggested by Hecker (2000) are based on the provision of open source software which is sold along with on-line services.

In the following section we briefly describe the characteristics of a subset of "Support Sellers", i.e. Linux distributors, and provide a preliminary analysis of the Red Hat business model. In Appendix E we report examples of other open source vendors classified into two categories: a) Linux distributors (Caldera, Turbolinux, and Suse); b) proprietary commercial software and hardware companies that have entered the OSS market with some of their products (i.e. Apple, IBM, Netscape, Sun, Zope).

5.4.2. Linux Distributors

The Linux operating system is composed of a kernel and a set of GNU programs. There are about 164 Linux versions, 145 of which are in English language and 126 are Intel compatible (www.linux.org/dist/list.html).

Most versions of Linux – for both clients and servers - can be downloaded from the web free of charge. However, downloading Linux from the web presents some disadvantages. First, users should have a high speed connection to Internet or a drive that writes on blank CDs in order to download fast and to store the software on an electronic medium. Second, if Linux has to be installed on a computer that already run another operating system the user has to be skilled in partitioning the hard disk. Finally, users may only have technical support from mailing lists provided by the community of developers, not

from vendors (www.linux.org/dist/download_info.html). New business companies have entered the market since the early 1990s to supply the Linux software in versions that are easier to install and manage than the free versions available on the web, and to provide support, documentation and services.

We report below some common features of the leading Linux distributors (Red Hat, Caldera, SuSe, Turbolinux), which account for the largest share of the market.

- Many Linux distributors allow users to download freely their own Linux version directly from the website, both in object code and in source code; they also sell Linux versions both for professional and for home users workstations and servers. They offer the following Linux products and services:
  a) Linux desktop versions for home users (i.e. Linux desktop/client/workstation) may also be freely available on CDs supplied in magazines. Manuals on the Linux operating system may also provide a free distribution on CDs. These 'commercial' distributions (even when they are freely distributed by magazines) are easier to install because the Linux main distributors include an 'installer' that facilitates the operation. The boxes sold are generally supplied with documentation; office packages; manuals and support services for a limited number of days (generally 60); a graphical interface (GNOME or KDE); software maintenance (i.e. for downloading and installing new releases, patches, new versions with bugs fixed).
  b) professional workstations for business companies or other institutions, which also include office applications and applications for LANs. Professional distributions are also supplied with more sophisticated support services compared with home versions.
  c) Linux server, a platform suitable for (among others) e-mail exchange, Internet and application services usually supplied with application packages for server and business applications. Further support is supplied with specific contracts that define both the kind of support the adopter will receive (for example installation and configuration support by phone or by web) and the way in which support is supplied (for example, Red Hat offers Monday to Friday 9am-9pm customer assistance with response in 4 hours or 24×7 with response in 1 hour). Moreover, maintenance services (i.e. for downloading and installing new releases, patches, new versions with bugs fixed) are supplied.

- Some distributors sell software internally developed for server and database management. This software can be both open source and proprietary, and can be based both on OSS and proprietary code. For example the Red Hat database integrates Red Hat Linux and PostgreSQL, while SuSe supplies a database server, distributed with the source code, based on the IBM DB2. Distributors

also supply platforms for multiple servers system management through a graphical interface, e-commerce suite, workflow management software, firewall software (Suse Linux Firewall is supplied with the source code).

- They sell high value added services, that is consulting, Internet connections, security strategies and database management. They also provide e-learning and Linux courses.
- They offer customised solutions (i.e. software solutions including planning, implementation, maintenance, support and training).
- There are differences across distributors in the share of software and services in total sales. For example, in 2001 software sales accounted for about 44% of Red Hat's revenues while Caldera drew over 84% of its revenues from software (data source: www.hoovers.com).

Recently Caldera, SuSE, Turbolinux and the Brazilian Conectiva have announced an alliance which aims to produce a common operating system kernel called UnitedLinux. The new distribution will comply with the Linux Standards Base (LSB), which is a project aiming at standardising Linux across the industry. This is an attempt to cope with the problem of inconsistencies among the Linux distributions and could make more easier for ISVs to develop applications running on Linux. According to Larry Seltzer[38], this agreement is likely to yield advantages for important vendors, like Borland, that are currently forced to adapt their product to different versions of the kernel and different versions of code libraries. As a matter of fact the alliance is supported by several companies including Borland, Computer Associates, and IBM[39].

The market strategy underlying the UnitedLinux agreement is not clear. As a matter of fact, the partners announced that they will develop a unique kernel but that each partner will continue to offer different versions of the UnitedLinux package (each containing different additional products and services) under their own brand name in different markets. Some argue that the main target of this agreement is to contrast the market power of Red Hat, the largest Linux distributor in the United States. However, the four partners reply that the agreement is open to all Linux distributors, including Red Hat and Mandrake[40].

In the next subsection we present a preliminary description of the Red Hat business model. Our further research on this issue will take into account more carefully Red Hat's market strategy and specific characteristics of its community of developers.

---

[38] Tech Update, June 19, 2002.
[39] Michelle Delio, Mired News, May 30, 2002 at www.wired.com/news.
[40] Michelle Delio, Mired News, May 30, 2002 at www.wired.com/news.

5.4.3. The case of Red Hat

Red Hat is the leading commercial supplier of Linux. It was founded in 1995 as Red Hat Software Inc. It went public in 1999. Its main products are high-end server product and development tools. Red Hat's staff is made of 300 engineers and includes the top 10 Linux kernel developers and seven of the top 10 open source development tools engineers[41].

- **Core business** Initially Red Hat based its core business on packaging and distributing Linux operating systems to end users, deriving revenues primarily from technical manuals and support. Now its focus is mainly on the web server market; furthermore, Red Hat is focusing on the provision of services, especially custom development for embedded applications, and new products such as new applications for database management and e-commerce. It also sell manuals; the company has partnered with Hungry Minds, Inc. to publish various publications on Linux or other open source products. These books are available for previewing online, as well as for purchasing usually through Amazon.com. A description of product and services offered by RedHat is presented in Table 5.5.

  We can place Red Hat in the space of R&D and commercialisation activities in the following way:

Activities	Basic / applied research	Development	Distribution	Services
Red Hat	-	X	XXX	XXX

Legend: X = of limited importance; XX = important; XXX = very important.

- **Relations with Open Source Community** Red Hat actively participates to open source software production. It has contributed to the development of several projects (Linux kernel and device drivers, Apache, Cygwin, GNOME, GNU Tools) and it is a member of many Linux associations (for example, Linux Standard Base to standardise the elements of Linux-based operating systems). Also, Red Hat provides web, FTP, and other Internet hosting services for several open source community projects (for example, GCC the GNU compiler collection)[42]. Red Hat hosts a Red Hat community mailing list and a Red Hat user group program (see http://www.redhat.com/apps/community/).

[41] Source: Dan Farber, Tech Update, June 20, 2002.
[42] Further exploration will aim to collect data on the number of developers actively participating to the open source software that Red Hat includes in its products.

- **Revenues** In 2000[43] Red Hat's revenues totalled about $103.4 millions (and decrease to 78.9 million in 2001) while gross profits were about $4.9 millions. Over 44% of revenues came from software subscriptions, 55% from services and 1% from other activities. Furthermore, 81% of the revenues were realised on the US market and 19% in other countries (data from www.hoovers.com, see Table 5.3 and Table 5.4). In 2001 47% of its revenues came from software subscriptions by enterprises and 7% from embedded software; 31% of revenues were from services related to open source software and 15% were related to services on embedded development. In addition, 70% of the revenues were realised on the US market, 15% in Europe and 15% in Asia Pacific and Japan.
- **IPR Strategy and Licensing Model** Red Hat packages contain several software elements developed by different sources (e.g., Sun Staroffice and Loki games in the Red Hat Linux 7.2 Standard Edition distributed in Europe). Red Hat licenses state that each component has his own end user license and that underwriters have to review the on-line documentation and comply with each licence terms. According to this policy, the Linux software is distributed under GPL. Red Hat license for Linux states that the underwriter has the right to transfer his copy of Red Hat Linux to other parties but only the original underwriter has the right to receive technical support. Despite the open source community position on software patents, Red Hat recently has applied for three software patents: two of them for methods and apparatus for atomic file lookup; and the third one for Embedded Protocol Objects (Source: www.Freego.it). Open source developers and companies such Red Hat usually argue that software patents hamper innovation, and today it very difficult to create software that do not use previous patented software (see for example http://www.fsfeurope.org/). Red Hat has also claimed that it is adopting the tactic to maintain a defensive patent arsenal, in case it need to launch a counter-suit for patent infringement. Also, Red Hat said it will not attempt to enforce its patents when they are used in open source software, except for software covered by BSD licence, since software covered by BSD can be integrated into proprietary software[44].

Table 5.3 Red Hat Sales by geographical area, 2000-2001

2000	$ mil.	% of total
North America	83.4	81
Other	20.0	19
Total	103.4	100

2002	$ mil.	% of total
North America	55.6	70
Asia Pacific & Japan	11.7	15
Europe	11.6	15
Total	78.9	100

Source:www.hoovers.com

[43] Red Hat set its fiscal year-end on February. So data on sales in 2000 are those declared in February 2001 and data on sales in 2001 are those declared in February 2002.
[44] Source: M. Broersma, ZDNet May 31, 2002.

Table 5.4 Red Hat Sales by product and services, 2000-2001

	% of total
Services	55
Software Subscription	44
Other	1
Total	100

Software Subscription	$ mil.	% of total
Enterprise	36.8	47
Embedded	5.5	7
Services		
Open Source	24.3	31
Embedded Development	12.3	15
Total	78.9	100

Source:www.hoovers.com

Table 5.5 Red Hat products

Free downloadable software (Source:www.redhat.com/apps/download)	▪ **Red Hat Linux 7.3** ▪ **Source navigator** a source code analysis tool for edit source and build project ▪ **RPM package manager** a tool that allows user to take source code for new software and package it into source and binary form
Linux Products (Source: www.redhat.com/products)	▪ **Red Hat Linux 7.3 Personal** for final end user ▪ **Red Hat Linux 7.3 Professional** workstation for professional use in small network and small business ▪ **Red Hat Linux for Itanium™ Processor** workstation designed for enterprise ▪ **Red Hat Linux Advanced Server (Standard, Premium and Advanced)** a server Linux; to different edition correspond different support services.
Other Red Hat Software	▪ **CMM Red Hat Content and Collaboration Management Software** Tools for management of complex activities in a business organisation (i.e. supply the complete set of task to perform, assign the tasks to worker, assign each task a deployment rule). *Red Hat CMM is delivered with the source code* allowing easy extensibility and customisation of the implementation by Red Hat, or by the organisation's own developers. Specific solutions for different companies are provided. In this sense this software can be viewed as a customise software ▪ **Red Hat Database** a database management system that integrates Red Hat Linux 7.1 and PostgreSQL 7.1.2 plus a Red Hat Installer ▪ **Red Hat E-Commerce Suite** Open source software solution that helps small and mid-sized businesses build and manage e-commerce applications. Includes several modules, including the Red Hat Linux Database and the Red Hat Linux 7.1 operating system. ▪ **Red Hat eCos vl_3** an open-source, royalty-free, configurable, operating system for embedded systems. It is targeted at applications in consumer electronics, telecommunications, automotive and other cost-sensitive and lightweight applications. ▪ **Red Hat Stronghold 3** Web server based on the open source Apache technology.
Other products	▪ Manuals; clothing and accessories.
Services	▪ Support, consulting, training, personalised solutions, custom development for embedded applications. *Note that many support services are supplied by contract with each packages distribution.*

5.5 Conclusions and policy implications

According to several scholars the OSS model represents a significant organisational innovation in the software industry and it is often referred to as an example for other industries as well (e.g. Kogut and Turcanu, 1999; Lerner and Tirole, 2000). A key characteristic of this model is represented by the modularity, which comes from the Unix environment. Modularity is enhanced by the use of the last generation programming languages (object-oriented languages such as C++ and Java). Another related characteristic is the distributed development. Task partitioning made possible by the modularity of process technologies and the diffusion of the Internet allow a rising physical distribution of innovative activities. In particular, a major organisational feature of OSS is associated with the possibility to develop a wide array of innovations (or product varieties) to meet heterogeneous users' needs. As Franke and von Hippel (2002) have illustrated in the case of the Apache web servers, OSS offers customers with heterogeneous needs the possibility to rely on a powerful and relatively easy-to-use toolkit for developing innovations that no commercial software supplier would have enough incentive to supply. This model of innovation, which is in part common to proprietary software (a distributed environment can be emulated by large firms and their networks of partners), is a potentially important source of innovation for this industry, which relies on lead users as a major source of innovation. The free access to the source code to a potentially large number of contributors can have positive effects on incremental technical change. In fact, however, we do not know how many people are interested in the source code. Probably a majority of users (including very sophisticated ones) are not interested in the source code as such but are more interested in freely available programs that can be used as a stand alone product or development tool.

Moreover, except for few cases such as Apache, Sendmail and Perl, OSS software often produces improvements of existing (often proprietary) software (e.g., StarOffice). Future empirical research should explore whether the OSS community is able to provide enough incentives to produce radical innovations or major departures from existing proprietary inventions. Advocates of proprietary software argue that OSS poaches on the pool of proprietary knowledge thus producing a negative externality (with negative consequences for the ex ante incentives to innovate). Advocates of OSS, on the contrary, reply that in the history of software innovation is a social construction or a combination of sequential unpatentable ideas. Moreover, the strength of the proprietary system can distort the incentives of the OSS developers and industry observers have claimed that patent examiners have granted too broad patent protection to applicants that have heavily relied on software developed in the OSS community (Lerner and Tirole, p. 30). Then potentially negative externalities are generated by both communities with negative net effects on innovation and growth. Future research in this field

should investigate the patterns of inventions generated in different OSS projects and ask what is the value of the knowledge disclosed for 'philanthropic innovators'.

Another issue is about the links with open science. We are interested to see whether OSS developers publish their results in scientific journals and patent their inventions. Moreover, future research should explore the coexistence of different regimes of property rights. As discussed before the two regimes analysed in this work tend to produce global negative effects on growth so long as they also compete in the market for software goods. A classical solution to this dilemma is to separate software R&D from software distribution. The R&D sector is obviously subject to market failure and as such should receive public support. In practice, public support to both 'philanthropic innovators' and profit oriented firms could be conditioned upon their pre-commitment to full or partial disclosure of the source code. On the other hand, in the distribution sector firms can appropriate the benefits of their inventions by relying on the available array of legal instruments, including patents. In order to favour the diffusion of patented software inventions, limited reverse engineering rights and access to the source code for experimental/research reasons could be introduced.

The advocates of OSS have emphasised the advantages of the large scale of voluntary developers and users who contribute to the improvement and debugging of products. However, with few exceptions few have addressed the issue of coordination costs associated with this model of innovation. Coordination costs in large scale projects can offset the advantages of distributed development. Čubranić and Booth (1999) discuss different solutions to the coordination problem adopted in different OSS projects. Most solutions rely on the division of tasks between leaders (who are usually the first developers) and a limited number of key developers/administrators who help the leaders in the scrutiny of new contributions, certification and integration of the new patches into the kernel. Despite the division of labour, the growing complexity of many OSS projects (e.g., Linux have reached over 1.5 million lines of code recently) results in longer release time (e.g. Linux version 2.2 took longer than expected and Mozilla took over one year to pass the official beta test). These coordination problems raise the question of scarcity. Raymond and his colleagues argue that OSS is a community founded on the abundance of resources. But the lack of coordination skills (and leadership), that are not easy to reproduce and to find in the market can become an important bottleneck that severely constraint the future growth of OSS.

Finally, the peculiar characteristics of the OSS business models described by Hecker (2000) and briefly analysed in section 5.4 highlight an important issue. Packaged software based on OSS products are often sold at very low price. For example, a Suse Linux 7.3 Professional Edition is sold at 68 Euro while Red Hat Professional 7.2 is sold at 211,67 Euro[45]. At the same time Wheeler (2001) computes

[45] Source: Ulrick B. and Zucchelli D., *PC Professionale*, January 2002.

that the cost to configure a server is about $1510 for installing the Windows 2000 operating system on 25 clients, and $156 for installing the Red Hat Linux operating system on an unlimited number of clients. Therefore some argue that the cost savings from the adoption of OSS can be fairly large.

However, the right way to evaluate the cost of adopting an OSS product is to look at the total cost of ownership (TCO). As a matter of fact, the adopter has to consider all costs involved in acquiring, installing, configuring, supporting, maintaining, using, and upgrading the software. These costs depend on the underlying assumptions used to build the cost model (Wheeler, 2001), the local technical and market environment in which the software is used, and the availability and cost of computer technicians with the necessary skills (Rusten and Moses, 2002). Also, TCO depends on the specific characteristics of the adopted product. It is not correct then to conclude that TCO for any OSS product is lower than for proprietary products. As far as we know, there are not reliable empirical data on the TCO of software, and to be informative, those data should be distinguished according to the type of user, i.e. individuals, enterprises or public administrations. For example, the decision of introducing OSS instruments in educational computer environments must be supported by a TCO specific model, which considers all costs associated with building the capacity of educators to integrate new instruments into teaching and learning activities. These costs might be very high when teachers have to be re-trained, when there is a lack of technical support so that small technical problems can prevent an effective use, or there is a lack of educational software applications that can operate on Linux (Rusten and Moses, 2002).

Our interviews suggest that OSS business actors have the perception that TCO is lower for open source products because the initial low price. However, our interviews also indicate that the estimated TCO of OSS is still high because of high level of skills required to manage and use the software, and the limited availability of complementary products (including documentation). But our interviews also point out that TCO for OSS is declining because of the introduction of GUI (graphical user interfaces), that reduce the level of skill required to use the software, and the availability of a huge amount of documentation on the web.

Appendix A

The OSI Approved Licenses

1. GNU General Public License (GPL)
2. GNU Library or 'Lesser' Public License (LGPL)
3. BSD license
4. MIT license
5. Artistic license
6. Mozilla Public License v. 1.0 (MPL)
7. Qt Public License (QPL)
8. IBM Public License
9. MITRE Collaborative Virtual Workspace License (CVW License)
10. Ricoh Source Code Public License
11. Python license (CNRI Python License)
12. Python Software Foundation License
13. zlib/libpng license
14. Apache Software License
15. Vovida Software License v. 1.0
16. Sun Industry Standards Source License (SISSL)
17. Intel Open Source License
18. Mozilla Public License 1.1 (MPL 1.1)
19. Jabber Open Source License
20. Nokia Open Source License
21. Sleepycat License
22. Nethack General Public License
23. Common Public License
24. Apple Public Source License
25. X.Net License
26. Sun Public License
27. Eiffel Forum License
28. W3C License
29. Motosoto License
30. Open Group Test Suite License
31. Zope Public License
32. University of Illinois/NCSA Open Source License

Appendix B

Free Software Foundation Approved Licenses

- **GPL-Compatible Free Software Licenses**

1. GPL
2. LGPL
3. The Guile License
4. The License of the run-time units of the GNU Ada compiler
5. The X11 license
6. Expat license
7. Standard ML of New Jersey Copyright License
8. The Cryptix General License
9. The modified BSD License (BSD license without advertising clause)
10. The Zlib License
11. The iMatix Standard Function Library License
12. The W3C Software Notice and License
13. The Berkeley Database License (Sleepycat Software Product License)
14. The OpenLDAP License, Version 2.7
15. The License of Python 1.6a2 and earlier versions
16. The License of Python 2.0.1, 2.1.1, and newer versions
17. The Perl License
18. The Clarified Artistic License
19. The Artistic License, 2.0
20. The Zope Public License version 2.0
21. The Intel Open Source License (as published by OSI)
22. The Netscape Javascript License

- **GPL-Incompatible Free Software Licenses**

1. The Arphic Public License
2. The original BSD license (with advertising clause)
3. The Apache License, Version 1.0
4. The Apache License, Version 1.1
5. The Zope Public License version 1
6. The license of xinetd
7. The License of Python 1.6b1 and later versions, through 2.0 and 2.1
8. The old OpenLDAP License, Version 2.3
9. The license of Vim, Version 5.7
10. IBM Public License, Version 1.0
11. Common Public License Version 0.5
12. The Phorum License, Version 1.2
13. The LaTeX Project Public License
14. The Mozilla Public License (MPL)
15. The Netizen Open Source License (NOSL), Version 1.0
16. The Interbase Public License, Version 1.0
17. The Sun Public License
18. The Nokia Open Source License
19. The Netscape Public License (NPL)
20. The Jabber Open Source License, Version 1.0
21. The Sun Industry Standards Source License 1.0
22. The Q Public License (QPL), Version 1.0
23. The FreeType license

24. The PHP License, Version 2.02

- **Non-Free Software Licenses**

1. The (Original) Artistic License
2. The Apple Public Source License (APSL)
3. The Sun Community Source License
4. The Plan 9 License
5. Open Public License
6. The Utah Public License
7. eCos Public License
8. The Sun Solaris Source Code (Foundation Release) License, Version 1.1
9. The YaST License
10. Daniel Bernstein's licenses
11. The 'Aladdin Free Public License'
12. The Scilab license

Appendix C
OSS Licence Models

The GPL

Source code availability	Yes. Source code has to be available for each distribution of the program or work based on the program. The code, or modified parts of it, can be distributed in executable form provided that the source code is available under the GPL terms.
Right to USE	No limitations. The act of running the program is not restricted, and the output from the program is covered by GPL only when the program copies part of itself into the output.
Right to COPY	No limitations.
Right to MODIFY	Yes, providing that the modified files carry a notice stating who has modified the program and the date of modification; each modification has to be distributed under the GPL terms. Note that who adopts the GPL-ed software is free to make modifications and use them privately without ever releasing them. But if the 'underwriter' decides to release the modified version, he or she has to make the modified code available under the GPL conditions.
Right to DISTRIBUTE	Yes. Any copy of the program and work based on the program have to be distributed under the GPL terms, that is distributors have to publish on each copy of the product the original copyright notice and the disclaimer of warranty; distributors have to give any another recipients the same rights to use, copy and modify that he or she has received. It is possible to distribute only executable code if a written offer to distribute the source code later is provided, and a price no greater than the cost required to physically distribute the source code has to be charged. As a special exception, the source code distributed does not need to include anything that is normally distributed as a component of the operating system on which the executable runs.
Moral rights protection	Yes. The GPL requires all copies of covered code to carry a copyright notice including the name of the copyright holder.
Initial developer 'special' rights	Yes. The original copyright holder may add an explicit geographical distribution limitation excluding those countries for which the distribution and/or use of the program is restricted either by patents or by copyrighted interfaces. In such cases, the licence incorporates the limitations 'as if' written in its body. Generally, the original copyright holder can release the software under different non-exclusive licences, including proprietary licenses. The original copyright holder can modify the terms for distributions of GPL covered software if an 'underwriter' asks for it. He or she can decide, for example, to allow the 'underwriter' to merge covered code with a proprietary product. But this can be done provided that the original code under the licence remains free, such that when other people modify the program, they do not have to make the same exception.
IS the software under the licence *gratis*?	Usually the software is free of charge but GPL does not ban sales, therefore distributors may sell copies of the *object code* and charge a fee for the physical act of transferring copies of the programme. The distributor may also offer warranty protection in exchange for a fee. Finally, he/she can charge a fee for the corresponding source code which cannot exceed the 'cost of physically performing source distribution' (see GPL Section 3)
Warranty / Liability / Claims	Disclaimer of warranty to the extent permitted by the applicable law. The 'underwriter' alone assumes the entire risk as to the quality and performances of the program and if the code proves defective, the 'underwriter' assumes the cost of all necessary servicing, repair or

	correction requested by subsequent users. The 'underwriter' may offer warranty protection in exchange for a fee. No liability, in no event unless required by applicable law or unless there is an explicit agreement with the copyrights holder or following distributors.
Third party claims	No explicit rules
National law compatibility	Compatibility problems could arise. For example, the 'viral clause' can conflict with copyright law when the author of modifications or integration of original covered code can demonstrate that those are original creations that rely only to a minimal extent to the original GPL covered code. The GPL claims that if in a given country it is not possible to distribute the program under conditions that guarantee the compliance with the obligations of the GPL and any other pertinent obligations (such as court judgement and allegation of patent infringement) then the 'underwriter' may exclude that country from the list of country where the distribution is allowed (GPL, Section 8).
Litigations / Conflicts / Violations	No explicit rules to solve conflicts are present. In case of violations of GPL terms, the user (or the company) looses the right the license gave him, but parties that have received copies of software or the rights from him/her will not have their licence terminated. Moreover, the Free Software Foundation strongly recommends to communicate them eventual violations, and to return back to them the rights of improvements any work based on the covered code in order to give the FSF the right to sue against infringements.
SPECIFIC CHARACTERISTICS OF OSS LICENCES	
'Mixability' with OSS licensed product	GPL is compatible with those licences for which is 'technically' possible to combine parts of GPL covered code with any module of code under a different licence in a larger work satisfying both licences at once. Generally, this applies to free software licences with copyleft property. For example GPL is compatible with the following licences: LGPL, Zlib licence, W3C, Sleepycat Software Product Licence, Python 1.6a2 and earlier version, Python 2.0.1, 2.1.1 and newer version, Clarified Artistic Licence, Zope Public Licence, Intel Open Source Licence.
'Mixability' with proprietary licensed product	No, unless the original copyright holder makes an exception.
Degree of openness protection	Maximum degree by copyleft property. Richard Stallman (GNU Organisation and Free Software Foundation) created the term 'copyleft' in opposition to copyright: with the term 'copylefted software' Stallman refers to free software whose distribution terms do not let distributors add any additional restriction when they redistribute or modify the software.

Source code availability	Yes. The source code has to be available for each distribution of the program or work based on the program. The code, or modified parts of it, can be distributed in executable form provided that the source code is available under the LGPL terms.
Right to USE	No limitations. The act of running the program using a LGPL-ed library is not restricted. The output from the program is covered only if its contents constitute a "work based on the library", that is a work containing the library or a portion of it, either verbatim or with modifications.
Right to COPY	No limitations.
Right to MODIFY	Yes, providing that a) the modified work is itself a software library; b) the modified files carry a notice stating who has modified the program and in which date; c) the whole work has to be licensed at no charge to all third parties under the terms of the licence; d) it is possible to add facilities which refer to functions or tables of data supplied by an application program, provided that if the application does not supply those functions or tables, the facilities still operate and perform the parts not requiring the functions or tables.
Right to DISTRIBUTE	Yes. LGPL distribution rules are quite complex. It is useful to distinguish among works based on the library and works that use the library. A "work based on the library" is a work containing the library or a portion of it, either verbatim or with modifications; furthermore, each work that is not derived from the library but is distributed as part of the whole which is a work based on the library, has to be considered as a work based on the library itself. A **work that uses the library** is code designed to work with the library in that it is compiled (or linked) with the library but does not contain portions of original library nor modified parts of it. The 'underwriter' of the LGPL licence can distribute the covered code or **works based on the library** provided that the whole work is licensed at no charge to all third parties under the terms of the LGPL licence. Complete source code has to be made available, that is distributors have to supply all the source code for all modules the library contains, any associated interface definition files, the scripts used to control compilation and installation of the library. Furthermore, each copy has to be accompanied by the copyright notice, a disclaimer of warranty, and a copy of the licence. Distribution of executable code without supplying the source code is allowed provided that the object code is accompanied by a written offer, valid for at least three years, to give the users the source code for a charge no more than the cost of performing the distribution. Generally a **work that uses the library**, in isolation, is not a derivative work and LGPL terms do not apply on it. But, the licence contains a list of exceptions and particular cases. For example, the object code of a work that uses material from a header file that is part of the library could be a work based on the library (the rules that distinguish a derivative from the covered code from a work that uses a library are not clearly defined by law). In that case the object code, or the executable that contains it, can be distributed provided that the source code of the ORIGINAL library and its modified parts is supplied. It is possible not to supply source code if the object code is accompanied by a written offer, valid for at least three years to give the users the source code for a charge no more than the cost of performing the distribution. For an executable, the required form of the work that uses the library must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed do not need to include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable. Furthermore, if the code is a work that is linked with the library in order to create an executable, that work has to be considered a derivative on the library. LGPL allows to

	distribute this work provided that both object and source code of this work are supplied in order to allow users to modify the work and re-link them whenever they need. It is allowed to use a suitable shared library mechanism for linking with the library instead of distributing source code. This mechanism uses at run time a copy of the library already present on the computer of the user and will operate properly with a modified version of the library if the user installs one. As an exception of distribution rules, it is allowed to combine or link a work that uses the library with the library to produce a work containing portions of the library and distribute that work under terms other than LGPL provided that the terms allow modifications of the code for customer's own uses and reverse engineering for debugging such modifications. Finally, section 7 states that it is allowed to place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by the LGPL, and to distribute such a **combined library**, provided that the separate distribution of the work based on the library and of the other library facilities is otherwise permitted, and provided that: a) the distributors accompanies the combined library with a copy of the same work based on the library, and not combined with any other library facility. This must be distributed under the terms of LGPL for work based on the library; b) it gives prominent notice with the combined library of the fact that part of it is a work based on the library, and explaining where to find the accompanying not combined form of the same work.
Moral rights protection	Yes. The LGPL requires all copies of covered code to carry a copyright notice including the name of the copyright holder.
Initial developer 'special' rights	See GPL
IS the software under the licence *gratis*?	Usually is free of charge but LGPL does not ban sales, therefore distributors may sell copies of the software. The distributor may offer warranty protection in exchange for a fee, or charge a fee for the physical act of transferring copies. When distributors supply only the executable code, they can provide the source code and charge a price no larger than the cost needed to perform the distribution.
Warranty/ Liability/ Claims	Disclaimer of warranty to the extent permitted by applicable law. The 'underwriter' alone assumes the entire risk as to the quality and performances of the program and if the code prove defective, the 'underwriter' assumes the cost of all necessary servicing, repair or correction requested by subsequent users. The 'underwriter' may offer warranty protection in exchange for a fee. No liability, in no event unless required by the applicable law or unless there is an explicit agreement with the copyrights holder or following distributors.
Third party claims	No explicit rules.
National law compatibility	See GPL
Litigations/ Conflicts/ Violations	No explicit rules to solve conflict are present. In case of violations of LGPL terms, the user (or the company) looses the right the license gave him but parties that have received copies of software or the rights from him/her, will not have their licence terminated. Moreover, the Free Software Foundation strongly recommends to communicate them eventual violations, and to returns back them the rights of improvements any work based on the covered code in order to gives the FSF the right to sue against infringements.
SPECIFIC CHARACTERISTICS OF OSS LICENCES	
'Mixability' with OSS licensed product	Yes, provided that the original library and its modifications are supplied with the source code. An executable which contains or uses original LGPL-ed library's code could be distributed under others licences agreements, provided that the user has the possibility and

	the right to isolate original library code from the executable and work on it applying the LGPL rules. An executable for which this rule is applied could be composed of (for example): code linked to LGPL-ed original library code; code that uses the library's header file; facilities that enriched original library; combination of LGPL library components and library under others license agreements.
'Mixability with proprietary licensed product	Yes, see the cell above.
Degree of openness protection	High degree. The licence imposes the 'copyleft' property on original library covered by the license.
Is the licence GPL-compatible?	Yes, if a library is under LGPL terms distributors may opt to apply the GPL terms on it, but once this change is made in a given copy, it is irreversible for that copy so GPL applies to all following copies and derivative works.

MPL: The Mozilla Public Licence

Source code availability	Covered code and any modification made by contributors must be available in source code. Covered code can be available in executable form under a different licence, providing that the source code is available. At the same time, the definition of 'modifications' does not apply to pieces of code that are not derived from the original code and are stored in a separate files.
Right to USE	No limitations.
Right to COPY	No limitations.
Right to MODIFY	Modifications are subject to the following conditions: 1) acknowledge first the author and the name of the original covered program; 2) distribute original source code except for added part of code not derived from original covered code; 3) distribute file-documenting changes. Furthermore, in MPL 1.1 contributors have to declare that they believe their modifications or improvement are original creations.
Right to DISTRIBUTE	Yes. MPL allows 'underwriters' to merge open source code with software distributed under other licence schemes (including proprietary ones) and to distribute also the executable version under other licence schemes (including proprietary ones) provided that they distribute the source code of the original covered software.
Moral rights protection	Yes. The 'underwriter' has to notify the name of the original developer and the original product name in each distribution and in every documentation.
Initial developer 'special' rights	In MPL 1.1 version the initial developer has the right to designate portions of covered code as "Multiple-Licensed". So that, the person, or society, whom the Initial Developer gives up her rights, can use portions of the covered code under alternative licence schemes, if any, specified by the initial developers.
IS the software under the licence *gratis*?	Yes, it is generally gratis but sale is allowed. The distributors may apply a fee to warranty protection and a fee in exchange for specific services (indemnity, support, and liability).
Warranty / Liability / Claims	Total disclaimer of warranty. The distributor alone assumes the cost of any necessary servicing, repair or correction. Limitation on liability except cases in which the applicable law prohibits such limitation. The distributor can offer, and charge a fee for, warranty, indemnity or liability obligations. This new provision does not affect the original covered code and the distributor agree to indemnify the original developer and contributors for any liability incurred as the result of the new grants that distributors offer.
Third party claims	**MPL 1.0** The licence is subject to third party intellectual property claims; third party claims have to be included in a LEGAL file. The distributor is responsible for damage arising, directly or indirectly, out of the utilisation of rights under the licence, based on the number of copies of code that made available, the revenues that received from utilising such rights, and other relevant factors. The 'underwriter' agrees to work with affected parties to distribute liability on equitable basis.
	MPL 1.1. Beyond the conditions concerning third parties claims mentioned before, each part is responsible for any damage arising, directly or indirectly, out of its utilisation of rights under the licence. The 'underwriter' agrees to work with the initial developer and contributors to distribute such responsibility on an equitable basis.
National law compatibility	If MPL is in conflict with national laws (statute and regulations), even about limitation on liability, the 'underwriter' has to comply with the terms of MPL as much as possible and describe limitations in a LEGAL file. Note: this section was included to allow to put under the licence also software, as cryptographic code, which may have legal restriction placed on its broad and public distribution.
Litigations / Conflicts / Violations	**MPL 1.0** The licences include terms for termination of the contract and a list of grants that will survive termination. The licences also specify which law - and with which exceptions - they are governed by and by which jurisdiction.

	MPL 1.1 The licences include terms for termination of the contract when there is a litigation about patent infringement claim and a list of grants that will survive termination. The licences also specify which law - and with which exceptions - they are governed by and by which jurisdiction.
SPECIFIC CHARACTERISTICS OF OSS LICENCES	
'Mixability' with OSS licensed product	Yes, MPL allows 'underwriters' to create a larger work by combining covered code with other not governed by the terms of the MPL and distribute it as a single product. In fact, this kind of licence was created in order to allow a company who usually distributes proprietary code to enter the open source world.
'Mixability with proprietary licensed product	Yes, see the cell above.
Degree of openness protection	High/Medium protection. Any modification made by contributors must be available in source code (so that the software remains free), but the definition of modifications does not apply to 'subroutines'. The MPL allows this new and separate piece of code to be distributed with covered code in a larger work under a different licence - even a proprietary one.
Is the licence GPL-compatible?	MPL 1.0 is not compatible with the GPL. In fact, MPL has a loophole that allows the 'underwriter' to make his/her own modifications private. MPL 1.1 allows in section 13 a software (or part of it) to be covered by another licence. If the 'underwriter' chooses the GPL, the two licences become compatible. Moreover, mozilla.org is providing the distribution of its product under a triple licence MPL1.1/GPL/LGPL.

The BSD licence

Source code availability	Yes.
Right to USE	No limitations.
Right to COPY	No limitations.
Right to MODIFY	No limitations
Right to DISTRIBUTE	Yes, distribution both in source and in object code are allowed provided that copyright notice and disclaimer of warranty is included in each copy distributed. Furthermore, underwriter has to be fully informed of the rights the licence gives him/her.
Moral rights protection	Yes, the first author and the contributors have to be mentioned in each copy of the licence. Some experts define this licensing type a "free marketing style" because programmers can show their skills to the business world simply giving away their product.
Initial developer 'special' rights	None.
IS the software under the licence *gratis*?	Generally gratis but there are not rules that ban sales and fees.
Warranty / Liability / Claims	The software is provided 'AS IS' without warranty of any kind; in no event shall the author or copyrights holder be liable for any claim, damages or other liability (except for countries where such clauses are in contrast with the law)
Third party claims	No explicit rules.
National compatibility	High compatible.
Litigations / Conflicts / Violations	No explicit rules.
SPECIFIC CHARACTERISTICS OF OSS LICENCES	
'Mixability' with OSS licensed product	Yes, The BSD licence model is highly permissive. The 'underwriter' can do almost anything with the software received.
'Mixability with proprietary licensed product	Yes. See the cell above.
Degree of openness protection	Low protection: source code distribution is allowed but not required.
Is the licence GPL-compatible?	Yes, except for the old BSD licence used until 1999. That version contained a clause requiring that any advertisement mentioning the BSD-software had to mention that the software was developed at the University of California. This additional clause is in conflict with the copyleft property of GPL. The Apache Licence 1.0 and 1.1 have still the same problem.

Appendix D

Diffusion of different license models

LICENCE	N°	%
Public domain	**856**	**3,11%**
Other/Proprietary licence	**549**	**1,99%**
OSI approved licenses	**26124**	**94,90%**
GNU General Public License	19096	69,37%
GNU Library or Lesser General Public License (LGPL)	2653	9,64%
BSD License	1812	6,58%
Artistic License	767	2,79%
MIT License	420	1,53%
Apache Software License	335	1,22%
Mozilla Public License 1.0 (MPL)	227	0,82%
Mozilla Public License 1.1 (MPL 1.1)	160	0,58%
Python License (CNRI Python License)	158	0,57%
Qt Public License (QPL)	139	0,50%
zlib/libpng License	129	0,47%
Common Public License	46	0,17%
IBM Public License	37	0,13%
Sun Industry Standards Source License (SISSL)	28	0,10%
Jabber Open Source License	20	0,07%
Apple Public Source License	17	0,06%
Nethack General Public License	16	0,06%
Intel Open Source License	11	0,04%
Python Software Foundation License	11	0,04%
Ricoh Source Code Public License	6	0,02%
Eiffel Forum License	5	0,02%
MITRE Collaborative Virtual Workspace License (CVW)	5	0,02%
Nokia Open Source License	5	0,02%
Sleepycat License	4	0,01%
Sun Public License	4	0,01%
Open Group Test Suite License	3	0,01%
University of Illinois/NCSA Open Source License	3	0,01%
W3C License	3	0,01%
Zope Public License	3	0,01%
Vovida Software License 1.0	1	0,00%
Motosoto License	0	0,00%
X.Net License	0	0,00%
TOTAL	**27529**	**100,00 %**

Source: elaborations on SourceForge (2002)

Appendix E
Examples of OSS vendors

A. Linux Distributors

Caldera

Caldera was founded in 1994 in the United States. In 2001 it bought the Unix products and services of Santa Cruz Operation (SCO) and became the latest owners of the 'official' Unix source code; after this acquisition it changed its name in Caldera International. In January 2002 Caldera decided to release some of the older UNIX versions under an open source license. Today Caldera provides complete solutions for development, deployment and management of unified Linux and Unix platforms. In 2001 its revenues were about 40.4 millions of dollars, 48% of which arising from sales in the US market, 37% in Europe, 11% in Asia, and 4% in Canada and Latin America (source: www.source.com). It sells open source based products (OpenLinux workstation and server, OpenUnix and SCO Open Server) and supplies support and services for those products. OpenLinux is distributed under GPL terms, and under the licenses of the other software integrated in the package[46].

Caldera offers customised solutions based on its products, support services for all major brands of Linux, consulting and training services. For example, Caldera OpenLearning sells Linux courses downloadable from their online stores for preparing the Linux certification exams and for other specific topics such as, for example, Linux system administration (www.caldera.com/education/courseware/). They also sell single lessons on specific topics (i.e. lesson on KDE desktop). Despite of the wide spectrum of services they provide, only 16% of sales are from services, while 84% are from software (Source: www.hoovers.com).

Caldera actively participates in free software production[47]. It has contributed to the development of several projects (Linux kernel, Java, Netscape, WordPerfect). Furthermore, Caldera supports several open source community projects (for example, Caldera Open Administartion System, Apache, KDE, Samba).

Table 1 shows a selected list of the main Caldera services and products.

[46] see for example www.caldera.com/support/docs/openlinux/1.3/english/license.html).

[47] see www.caldera.com/developers/community

Table 1 Caldera software and services

- **Selected Products**
  - OpenLinux (fully integrated Linux operating system)
  - OpenServer (UNIX-based server software)
  - OpenUnix (UNIX-based business application)
  - Volution (Web and directory-based network management)
- **Selected Services**
  - Consulting
  - Custom Engineering
  - Project Management
  - Technical support
  - Training

Source: www.hoovers.com

SuSe

SuSE is a leading European Linux distributor providing operating systems and application software for both individuals and corporations. It was founded in 1992 and has received financial spport from Silicon Graphics, Inc., IBM, and Intel (www.hoovers.com). Suse's enterprise packages contain applications for servers, data exchange, and multimedia functions.

The company has recently expanded its product line providing – among others - the SuSE Linux Firewall both in object and source code. Open source products are also freely downloadable from SuSE websites. It is possible to dowload several versions of SuSE Linux (i.e. i386, PowerPC, AXP, SPARC, S/390); updates, patches and versions with bugs fixed for all SuSE Linux version; KDE and GNOME packages, language support, applications and development packages. Finally, SuSE sells gadget as T-shirts and fashion accessories. SuSE provides also support services mainly available online (see for example http://support.suse.de/psdb/ for online support provided to business customer). Furthermore, SuSE offers consulting services and integrated solutions for business (i.e. planning, software implementation, maintenance and services).

SuSE Linux software is distributed under the GPL license. However, since the SuSE Linux package usually contains third parties software, each software is accompanied by a specific licence.

Table 2 shows a selected list of SuSE products.

Table 2 SuSe software and services

- **Selected Products**
  - Linux e-mail server
  - Linux enterprise server
  - Linux firewall
  - Linux groupware server
  - Linux database server
  - Linux network server

Source: www.hoovers.com

Turbolinux

Turbolinux Inc. was founded in 1992 in the United States and has received backing from August Capital, Dell, Intel and Fujitsu. Moreover, Oracle and IBM are among its key partners (data from www.hoovers.com). Turbolinux is the Linux leading supplier in Asia Pacific. In China Turbolinux is also contributing to the transformation of the digital infrastructure of the country.

The company provides Linux operating systems for clients and servers – such as Turbolinux 7 Workstation and Turbolinux 7 Server – and software for data-server that work on IBM DB2 and Oracle. It also provides PowerCockpit as software for "provisioning" (i.e. deploying and changing computing assets in a network) UNIX, Linux and Windows servers on a single platform. PowerCockpit seems to be the key competitive advantage of Turbolinux with respect to the other Linux distributors (www.turbolinux.com/about). In addition, Turbolinux provides clustering technologies like TurboLinux EnFuzion - a software that clusters (i.e. shares operations on multiple machines) all available computing resources on a company network to create a "virtual supercomputer" - and Turbolinux Cluster Server 6 for managing clusters. The company also provides support services (including free user-browseable documentation, FAQs, and package updates online) to business customer, consulting services and education.

Turbolinux is also supported by a community of users and developers. It invites the community members to test its beta version of software (http://www.turbolinux.com/beta/) and allows the users to participate to the development process by posting their contributions by ftp. Turbolinux is involved in the Linux-SNA open source project (http://www.linux-sna.org/) that has the aim to bring SNA - a set of network protocol developed by IBM – to Linux (www.turbolinux.com/devzone).

Table 3 shows a selected list of Turbolinux products and services.

Table 3 Turbolinux software and services

- **Selected Software**
  TurboLinux Cluster Server (integrates Linux, Windows, and UNIX servers)
  TurboLinux EnFuzion (uses existing computing resources for intensive calculations)
  TurboLinux DataServer with IBM DB2 (for running IBM databases on Linux)
  TurboLinux DataServer with Oracle 8i (for running Oracle databases on Linux)
  TurboLinux 7 Server (Linux operating system for servers)
  TurboLinux 7 Workstation (flagship operating system software for PCs and workstations)
- **Services**
  Customer support
  Product optimization and engineering
  Training

Source: www.hoovers.com

B. OSS and traditional commercial companies

Several traditional commercial software companies are developing open source software or releasing the source code of their products. Other hardware or software companies are supporting the development of open source software for their hardware or software platforms. In this section we briefly describe few examples of these companies which illustrate how *hybridization strategies* discussed before work in practice. Our future research will focus more deeply on these strategies and the implications of coexistence of proprietary and open source licensing models.

Netscape Communications

Founded in 1994 in the United States. Netscape is an Internet pioneer and a leading provider of web browsers. Its Navigator held 85% of market shares before the entry of Microsoft Internet Explorer in the client browser software market. Reacting to Microsoft threat in 1998, Netscape announced its intention to release its client software, including Netscape Communicator and Netscape Navigator, as open source software. At the same time, they created the Mozilla Organisation and the mozilla.org web site for stimulating developers to collaborate on this project. Netscape then released Communicator 4.5 with a 10 billions of dollars marketing campaign. America On Line (AOL) acquired Netscape in March 1999 (data from www.hoovers.com).

Today Netscape, as a division of AOL Time Warner, draws on open source Mozilla technology for its Netscape communication commercial browser. Mozilla software is also used in other open-source browsers, such as the Galeon browser for Linux.

The Mozilla.org members actively supports the browser development. They provide technical and architectural directions for the project, help authors to synchronise their work, and periodically release new source code by incorporating the best contributions. They also coordinate discussion forums (mailing lists, newsgroups, or other appropriate groups) and bug lists. In addition Mozilla members keep track of works in progress and advertise it; they also suggest improvements to the code and point to projects based on that code. It is worth to note that most code approved and distributed by Mozilla.org is written by Netscape engineers and other organisations in the net (see http://www.mozilla.org/mission.html). Netscape Communicator is distributed under the Netscape Public License (NPL) while Mozilla software is distributed under the Mozilla Public License (MPL).

Zope Corporation

Founded in 1995 in the United States (Fredericksburg, VA), Zope Corporation (www.zope.com) provides high-end custom solutions for big business companies (mainly media/telecommunication firms, newspapers, Internet business and Fortune 1000 companies) and public institutions (i.e., government, military, educational institutions). The company provides consulting services, web site

hosting services, technology partnerships, support and training. Its main product is the Zope application server, that enables the teams to collaborate in the creation and management of web based business applications such as portals.

In 1997 Zope corporation released its software as open source. Today, Zope source code is developed by the Zope Community (www.zope.org). There are no licensing fee for downloading and using the software; at the same time Zope sells personalised solutions of its software and provides con sulting services, technology partnerships, support, training and custom vertical applications. However , customers are not dependent on particular vendors for supporting and bug fixing because they can always find another supplier in the community[48].

In 2000 Zope acquired PythonLabs, which produces the open source object-oriented language Python (www.python.org). Today the Zope and Python communities are strictly integrated (http://www.pythonandzope.com).

The Zope software is distributed under the Zope Public License (ZPL) approved by the Open Source Initiative (OSI) board in February 2002 (http://www.opensource.org).

Apple Computers

Founded in 1976, Apple Computers, Inc. supplies desktop computers (iMac) and laptops (iBook) based on processors produced by IBM and Motorola. Its targets are high-end consumers and professionals involved in design and publishing. It also provides multimedia and publishing software and offers Internet services such as web pages hosting. It holds a majority stake in FileMaker, a producer of database software. Apple was the first traditional computer company to enter the open source community. It has contributed in several open source projects mainly related to Mac OS X (http://developer.apple.com/darwin/projects/). For example, in 1999 Apple released the core of Mac OS X server as open source called Darwin (http://www.opendarwin.org/) and distributed it under the BSD license; today software produced in Darwin project are distributed under the Apple Public License (http://developer.apple.com/darwin/licensing.html). It has also put open source the Quicktime Streaming Server and the OpenPlay network gaming toolkit (www.opensource.org/docs/products.html). Today it collaborates with the Apache Group and other open source developers for the development of the Mac OS X platform.

In 2001 its revenues were about 5,363 million of dollars, 25% of them from sales to schools (data from www.hoovers.com). Apple open source software (Darwin, Streaming Server, CDSA, OpenPlay,

[48] http://www.zope.com/Corporate/CompanyProfile .

HeaderDoc, and documentation on projects) are distributed under the Apple Public License (see http://developer.apple.com/darwin/projects/).

Apple supports a Darwin community whose members voluntarily contribute to develop software for the core operating system of Mac OS X, and for the Streaming Server.

IBM

IBM is one of the world's largest computer company. It produces software, PCs, mainframe and server systems, notebooks, microprocessors, and peripherals. It also supplies services including consulting, web hosting and training. In 2001 its service business accounted for about 40% of its total sales (data from www.hoovers.com). In the last years IBM has reorganised its hardware business, by merging its desktop and laptop operations and concentrating on its leading enterprise server and storage products. Today its focus is mainly on e-commerce infrastructures, databases, messaging and server software while it is moving away from operating system (www.hoovers.com).

This strategic repositioning was also characterised by the decision to strongly support the Linux development community. IBM has been contributing to many open source projects (see http://www-106.ibm.com/developerworks/). IBM has made strong interoperability efforts to allow its hardware and software products to work with Linux operating system and currently participates in Linux Internationalisation Initiative. It develops drivers to allow its hardware work with Linux; IBM is also improving and extending Linux for its hardware (i.e., Linux for S/390 and zSeries; www.ibm.com/S/390/linux). In addition, IBM has disclosed the source of an open source journaled file system technology (JFS), distributed under GPL terms, a key product for running intranet and other high-performance e-business file servers. IBM is offering this technology to the Linux open source community with the hope that it will be adapted to the Linux operating system (http://www-124.ibm.com/developerworks/oss/jfs/).

At the same time IBM has donated software to the open source community, mainly software programming tools, to spur the creation of applications for e-business and web services on the Linux operating system. Through the organisation Eclipse (whose members include Red Hat and SuSE) IBM donated the Eclipse-based tools that run on both Linux and Windows and allow developers to create a single application rather then first create the software on Windows and then transferring it to Linux[49]. In addition, IBM donated the software code for a Java application programming interface (a technology called UDDI for Java) that connects business actors to the giant online "Yellow Pages" that was created by Microsoft and IBM. Several companies are now supporting the UDDI for Java open-source project (for example Compaq, Bowstreet, Cross, DataChannel) that is a key technology for building a

[49] Source: Wong W., at http://zdnet.com.com/2100-11-527550.html, Update January 24, 2001.

system in which customers will not have to buy and install software on a personal computer, but will able to download what they need over the Internet[50] (as already Microsoft technology .Net allows to do).

IBM hosts open source projects on its web site also providing tools for active community members participation. We have not complete data on how each of these communities is organised. As an example, for the JFS project IBM provides a downloading area, the CVS repository to track all members activities on source code, documentation, mailing lists, and an area for reporting bugs. The JFS project is managed by a small, core group of contributors known as the JFS core team. They have read-write access to the JFS source code repository, decide which features go into which releases, decide who has access to the JFS source code repository, approve the check-in of new or changed source code and documentation, nominate new members of the core team.

IBM open source products are distributed under various licensing terms. They are often distributed under IBM Public License and Common Public Licence, but Linux software development is usually distributed under GPL terms.

Sun Microsystems

Sun Microsystems, founded in 1982, is the leading distributor of UNIX-based servers; it also provides computers – from desktop workstations to high-end servers – that run with Sun SPARC chips and Sun Solaris operating systems. The company also provides application servers, suite offices, data storage equipment, and network management software. In addition Sun offers for support, consulting and training services. Since 1995 Sun focused on the Internet computing, developed the Java programming language and introduced the Java-based network. Today, the Java source code is available to users that want to examine and modify the code, but it is not open source since developers cannot distribute it. In 1999 Sun formed with Netscape and AOL a joint-venture called iPlanet to develop e-commerce software. The join-venture with AOL expired in 2001 and iPlanet was then absorbed by Sun (www.hoovers.com).

In 1999 Sun began to participate in several free and open source projects (see www.sunsource.net). Since 2000 Sun is also providing server machines running Linux. It participates in Linux-related open source projects; for example it is working to guarantee the compatibility between the Sun Solaris operating system and Linux. Recently Sun has announced a free Platform Edition of its new Sun ONE Application Server 7 that allows to build commercial web sites (Eric Knorr, ZDNet June 21, 2002).

---

[50] Source: Junnarkar S., at http://zdnet.com.com/2100-1104-275388.html, Update November 5, 2001.

Sun has launched several open source communities. The Sunsource.net (www.sunsource.net), founded by Sun, allows users to participate to its open source product development. Communities members interact through the CollabNet platform, a web based development environment that enables geographically dispersed groups of developers to collaborate. Also, Sun supports a discussion list for general discussions about free and open source software, Sun policies and its use of open source licenses. In 2000 Sun founded the Netbeans.org – for open source development tools - and the OpenOffice.org for office productivity applications. In 2001 Sun launched a community on its Jxta, a project for distributed computing protocols.

Sun contributes to open source projects through its engineers working on several projects, such as OpenOffice.org, GNOME.org, Mozilla.org, Jxta.org, and Apache.org. It disclosed StarOffice code through OpenOffice.org and the code of Forte for Java IDE – cross-platform compatible development tools that enables the development on the Solaris Operating Environment, Linux and Windows platforms - through NetBeans.

Sun enjoyed several benefits by supporting open source projects. Several Sun commercial software including StarOffice and Forte is based on software developed by OSS communities. Recently Sun has stopped free downloads of StarOffice v 5.2 to sell StarOffice 6.0, making proprietary the software based on OpenOffice (M. Broersma ZDNet, May 27, 2002 at http://zdnet.com.com/2100-1104-923039.html).

Sun distributes its products under different licenses schemes. Open source products are distributed under GPL, Mozilla, or BSD terms, and also under traditional proprietary terms. OpenOffice is distributed under a dual licence strategy, that is i) GPL, which has the aim of receiving the contribution of the open source community; and ii) Sun Industry Standard Source License (SISSL) which is aimed at allowing commercial partners to utilise the technology within their product and/or provide branded versions of the technology.

References

Aoki A. *et al.* (2001), "A Case Study of the Evolution of Jun: an Object-Oriented Open-Source 3D Multimedia Library", *IEEE Software Engineering*, pp. 524 –533.

Bohem B. W. (1981), *Software Engineering Economics*, Prentice-Hall, Edgewood Cliffs.

Cohen J. E., Lemley M.A. (2001), "Patent Scope and Innovation in the Software Industry", http://www.georgetown.edu.

Cowan C. (1998), "Automatic Detection and prevention of Buffer-Overflow Attacks", *Proceedings 7th Usenix Security Symposium*, Usenix, San Diego, pp. 63-78.

Cŭbranić D., Booth K. (1999), "Coordinating Open Source Software Development", *IEEE Proceedings*, 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, (WET ICE '99), pp. 61–66.

Cusumano M. (1991), *Japan Software Factories. A Challenge to the US Management*, Oxford University Press, New York.

Dalle J., Jullien N. (2001a), "Libre Software: Turning Fads into Institutions?", http://opensource.mit.edu/online_papers.php.

Dalle J., Jullien N. (2001b), "Open-source vs. Proprietary Software", http://opensource.mit.edu/online_papers.php.

Dasgupta P., David P.A. (1987), "Information disclosure and the Economics of Science and Technology", in: G. Feiwel (ed.), *Arrrow and the Ascent of modern Economic Theory*, New York University Press, New York.

Dasgupta P., David P.A. (1994), "Towards a new economics of science", *Research Policy*, Vol. 23, pp. 487-521.

David P. (1991), "Reputation and Agency in the Historical Emergence of the Institutions of "Open Science", Stanford University, CA, March, mimeo.

Di Bona C., Ockman S., Stone M. (eds.) (1999), *Open Sources: Voices from the Open Source Revolution*, O'Reilly: Sebastopol, California.

EITO (2001), European Information Technology Observatory.

Franke N., von Hippel E., (2002), "Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software", *MIT Sloan School of Management Working Paper # 4341-02*.

FSF (2002), *Categories of free software*, http://www.gnu.org/philosophy/categories.html.

FSF (2002), *Licenses*, http://www.gnu.org/licenses/licenses.html.

FSF (2002), *Philosophy*, http:// www.gnu.org/philosophy.html.

Fuggetta A. (2001), "Open Source Software: an evaluation", Politecnico di Milano, mimeo.

Godfrey M.W., Tu Q. (2000), "Evolution in Open Source software: A Case Study", *IEEE Software Maintenance*, pp. 131 –142.

Grossman G., Helpman E. (1991), *Innovation and Growth in the Global Economy*, MIT Press, Cambridge, Mass.

Hamerly J., Paquin T., and Walton S. (1999), "Freeing the Source - The Story of Mozilla", in Di Bona C., Ockman S., Stone M. (eds.) (1999), *Open Sources: Voices from the Open Source Revolution*, O'Reilly: Sebastopol, California.

Hars A., Ou S. (2001), "Working for Free? – Motivations of Participing in Open Source Projects", *IEEE Proceedings* , 34th Annual Hawaii International Conference on System Sciences, pp. 2284 – 2292.

Hecker F. (2000), "Setting Up Shop: The Business of Open-Source Software", http://www.hecker.org/writings/setting-up-shop.html.

IDC (2000), *Server Operating Environment: 2000 Year in Review*, Bulletin#23731.

IDC (2002a), *Worldwide Integrated Collaborative Environments Forecast 2002-2006*, Bulletin#26579.

IDC (2002b), *Worldwide Web Server and Web Server Acceleration Software Forecast 2002-2006*, Bulletin#26519.

IDC (2002c), *Worlwide Software Support Services Forecast and Analysis 2001-2006*, Bulletin#27177.

Kargon R., Leslie S. W., Schoenberger E. (1992), "Far Beyond Big Science: Science Regions and the Organization of Research and Development", in Galison, P. and Hevly, B. (eds.), *Big Science. The Growth of Large-Scale Research*, Stanford University Press, ch. 13, pp. 334-354.

Koch S., Schneider G. (2000), "Results from Software Engineering Research into Open Source Development Projects Using Public Data", Vienna University of Economic and BA, http://opensource.mit.edu/online_papers.php.

Kogut B., Turcanu A. (1999), "Global Software Development and the Emergence of E-Innovation", Carnegie Bosch Institute, Carnegie Mellon University, Pittsburgh, mimeo, October.

Krishnamurthy S. (2002), "Cave or Community? An Empirical Examination of 100 Mature Open Source Projects", First Monday, Vol. 7, n. 6, http://firstmonday.org/issues/issue7_6/.

Jeppesen L.B. (2001), Making Consumer Knowledge Available and Useful. The Case of the Computer Games, Department of Industrial Economics and Strategy, Copenhagen Business School, mimeo.

Lakhani K., von Hippel E. (2000), "How open Source software works: 'Free' user-to-user assistance", MIT Sloan school of Management WP No. 4117.

Laing D. (1999), "Is GNU/Linux commercially viable?", http://www.smuts.uct.ac.za/~dlaing/essay/is_linux_commercially_viable/Is_Linux_Commercially_Viable.html.

Lehman M.M. (1985), *Program Evolution: Processes of software Change*, Academic Press.

Lerner J., Tirole J. (2000), "The Simple Economic of Open Source", NBER Working Paper n. 7600 (http://www.nber.org/papers/w7600).

Lerner J., Tirole J. (2001), "The Open Source movement: Key research questions", *European Economic Review*, Vol. 45, pp. 819-826.

McKusick M.K. (1999), "Twenty Years of Berkeley Unix - From AT&T Owned to Freely Redistributable", in Di Bona C., Ockman S., Stone M. (eds.) (1999), *Open Sources: Voices from the Open Source Revolution*, O'Reilly: Sebastopol, California.

Merges, R. (1996) "A Comparative Look at Property Rights and the Software Industry", in D. Mowery (a cura di) *The International Computer Software Industry. A Comparative Study of Industry Evolution and Structure*, Oxford University Press, Oxford, pp. 272-303.

Mockus A., Fielding R.T, Herbsleb J. (2000), "A Case Study of Open Source Software Development: The Apache Server", *IEEE Software Engineering*, pp. 263 –272.

Murtha R. (1998), "Open Source Software (OSS): A NewBusiness Paradigm?", http://is.gseis.ucla.edu/impact/f98/Projects/murtha/.

Netcraft (2001), *Web Server Survey Methodology*, http://www.netcraft.com/Survey/index-200007.html#active.

Netcraft (2001), *Operating System Survey Methodology*, http://www.netcraft.com/Survey/index-200109.html#computers2001.

Netcraft (2002a), *Web Server Survey*, http://www.netcraft.com/survey/.

Netcraft (2002b), *Web Server Operating Systems in Italy*, Netcraft for Microsoft.

Netcraft (2002c), *Netcraft Primer V1.1 – March 2002*.

OSI (2002), *OSI Approved Licenses*, http://www.opensource.org/license/index.html.

OSI (2002), *The Open Source Definition*, http://www.opensource.org/docs/definition.html.

OSI (2002), *The Open Source History*, http://www.opensource.org/history.html.

Perens B.(1999), "The Open Source Definition", in Di Bona C., Ockman S., Stone M. (eds.) (1999), *Open Sources: Voices from the Open Source Revolution*, O'Reilly: Sebastopol, California.

Prusa T.J., Schmitz J.A. (1991), "Are New Firms and Important Source of Innovation?", *Economic Letters*, 35; 339.

Raymond E. S. (1999a), "The Revenge of Hacker", in Di Bona C., Ockman S., Stone M. (eds.) (1999), *Open Sources: Voices from the Open Source Revolution*, O'Reilly: Sebastopol, California.

Raymond E.S. (1999b), "Linux and Open-Source Success", *IEEE Software*, Vol. 16, pp. 85-89.

Raymond E. S. (2000), "Homesteading the Noonsphere", http://tuxedo.org/~esr/writings/homesteading/homesteading/.

Raymond, E. S. (2001), *The Cathedral and the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly, www.o.reilly.com.

Rosenberg N. (1990), "Why Do Firms Do Basic Research (with their Own Money)?, *Research Policy*, 19, 165-174.

Rusten E., Moses K.D. (2002), "Open Source Software: No Free Lunch?", *TechKnowLogia*, January-March 2002, pp. 75-79, http://www.TechKnowLogia.org.

Saint-Paul G. (2001), "Growth effects of non-proprietary innovation", CEPR Discussion Paper Series No. 3069, http://cepr.org/pubs/dps/DP3096.asp.

Security Space (2002), *Security Space Web Server Survey*, April, https://secure1.securityspace.com/s_survey/sdata/200203/index.html.

Sirmi (2001a), *Il mercato Linux 1999-2003*, Sirmi for Microsoft.

Sirmi (2001b), *Mercato Linux*, June 2000, Sirmi for Microsoft.

Sirmi (2002), *Linux Survey*, January 2002, Sirmi for Microsoft.

SourceForge (2002), www.sourceforge.net.

Stallman R. (1999), "The GNU Operating System and the Free Software Movement", in Di Bona C., Ockman S., Stone M. (eds.) (1999), *Open Sources: Voices from the Open Source Revolution*, O'Reilly: Sebastopol, California.

Steinmueller W.E. (1996), "The U.S. Software Industry: An Analysis and Interpretative History", in Mowery D. (ed.), *The International Computer Software Industry: A Comparative Study of Industry Evolution and Structure*, Oxford University Press, Oxford.

Torrisi S. (1998), *Industrial Organisation and Innovation. An International Study of the Software Industry*, E. Elgar, Cheltenham.

Total Romtec (2001), *University Market Survey*, October 2001, Total Romtec for Microsoft.

Vixie P. (1999), "Software Engineering", in Di Bona C., Ockman S., Stone M. (eds.) (1999), *Open Sources: Voices from the Open Source Revolution*, O'Reilly: Sebastopol, California.

Viega J. *et al.* (2001), "A Static Vulnerability Scanner for C and C++ Code", http://www.cigital.com/papers/download/its4.pdf.

Von Hippel E. (1988), *The Source of Innovations*, MIT Press.

Von Hippel E. (2001), "Learning from Open Source Software", Sloan Management Review, Summer, 2001.

Wheeler D. (2002), "Why Open Source Software/Free Software (OSS/FS)? Look at the Numbers!", http://www.dwheeler.com/oss_fs_why.html.

Wilson G. (1999), "Is the Open-Source Community Setting a Bad Example?", *IEEE Software*, Vol. 16, pp. 23 –25.

Witten B., Landwehr C., Caloyannides M. (2001), "Does Open Source Improve System Security?", *IEEE Software*, Vol. 18, pp. 57 –61.

Zoebelein (1999), http://www.leb.net/hzo/ioscount, (ref. http://dwheeler.com/oss_fs_why.html).