

POLITECNICO DI MILANO

Faculty of Engineering

Masters of Science in Engineering of Computing Systems

Department of Electronics & Information



The Impact of business models on the performance of software firms'-the case of Open-source.

Supervisor: Assistant Professor Evila Piva

A Master's Thesis by:
MD.JASHIRUL ABEDIN CHOWDURY
Matr. 752851

Academic Year 2010 – 11

The Impact of business models on the performance of software firms'-the case of Open-source.

A Postgraduate Dissertation presented to

DEPARTMENT OF ELECTRONICS & INFORMATION

As a Partial Fulfillment for the Award of Degree

Master of Science in Engineering of Computing Systems

By

MD.JASHIRUL ABEDIN CHOWDURY

(Matriculation No. 752851)

Supervised by

Evila Piva

Assistant Professor, Dept of Management, Economics and Industrial Engineering

POLITECNICO DI MILANO

MILAN, ITALY

DECEMBER 2011

DEDICATIONS

To my family

- Jashir

ACKNOWLEDGEMENTS

Thanks to Allah for His blessings.

Thanks are in order to Prof. Cristina Rossi Lamastra and Asst. Prof. Evila Piva;

For their sincere guidance, support and efforts through all the time

Special thanks to POLIMI- it's an honor to be a part of it

Abstract

This thesis analyzes the impact of business strategies on the performance of software firms that have entered the open source software (OSS) field. The notion of the OSS business model is discussed in the light of a substantial body of theoretical literature concerning strategic management and the economics of innovation, as well as specialized literature on OSS. This empirically oriented research uses a unique data from the software industries of five European countries (Finland, Germany, Italy, Portugal and Spain) to illuminate these patterns of private, entrepreneurial provision of software placed in the public domain. The database comprises information from 915 European software firms of which 378 (or about 39%) were the suppliers of the Open Source Software (OSS) solutions at the time of the survey, the end of the year 2004. This thesis examines the stability of business models over performance in the evolution of the industry.

Astratto

Questa tesi analizza l'impatto delle strategie di business sulle performance delle imprese di software che sono entrati nel software open source (OSS) di campo. La nozione di modello di business open source è discusso alla luce di una quantità sostanziale di letteratura teorica sulla gestione strategica e l'economia di innovazione, così come la letteratura specializzata in OSS. Questa ricerca empiricamente orientata utilizza un dato unico dalle industrie del software di cinque paesi europei (Finlandia, Germania, Italia, Portogallo e Spagna) per illuminare questi modelli di privati, imprenditoriali fornitura di software messo di dominio pubblico. Il database comprende informazioni da 915 aziende di software europee, di cui 378 (pari a circa il 39%) sono stati i fornitori del Software Open Source (OSS) soluzioni al momento del sondaggio, la fine del 2004. Questa tesi prende in esame la stabilità dei modelli di business in più di prestazioni l'evoluzione del settore.

Table of Contents

Abstract.....	1
Table of Figure.....	5
List of Tables	6
1 Introduction.....	7
1.1 Motivation.....	8
1.2 Goal and Objective.....	8
2 History of Open-Source Software.....	9
2.1 The Analytic Problem of Open Source	9
2.2 Definition of Open Source Software.....	11
2.3 A brief history of open source software:.....	12
2.3.1 The University of Berkeley and the development of UNIX	12
2.3.2 The birth of Linux.....	13
2.3.3 The growth of the Apache web server group.....	15
2.3.4 Netscape- Rise, fall and possible recovery?.....	17
2.4 OSS Literatures	18
Summary.....	19
3 Economics of open-source software	20
3.1 The Economic Motivation of Open Source Software.....	20
3.2 SYSTEM INTEGRATOR PERSPECTIVE.....	21
3.2.1 IT solutions demand curve.....	21
3.2.2 Business growth.....	23
3.2.3 Pressures in the IT value stack.....	25
3.3 SOFTWARE VENDOR PERSPECTIVE.....	25
3.3.1 Software cost and pricing.....	25
3.3.2 Generating software profits.....	28
3.3.3 Commercial Open Source	28
3.3.4 Open source service firms.....	29
3.4 An Overview of Open Source Business & Community.....	30
3.4.1 Producers of Open Source Products—the Community.....	32

3.5 Business Models.....	34
3.5.1 The Distributor.....	35
3.5.2 The Software Producer (Non-GPL Model).....	36
3.5.3The Software Producer (GPL Model).....	38
3.5.4 The Third-Party Service Provider	39
4 Data Gathering.....	44
4.1 Data	44
5 Methodology of the econometric analysis	48
5.1. The specification of the econometric model	48
5.2. Dependent variables	49
5.3. Explanatory variables.....	50
5.4. Econometric results	52
6 Conclusion	56
Bibliography	58
Appendix.....	62

Table of Figure

Figure 2.1: Market Share for Top Servers Across All Domains (www.netcraft.com) 15

Figure 2.2: Totals for Active Sites Across All Domains June 2000 - October 2011(www.netcraft.com)..... 16

Figure 2.3: Market Share for Top Servers Across the Million Busiest Sites September 2008 - October 2011(www.netcraft.com) 16

Figure 3.1. IT solutions demand curve 22

Figure 3.2 Sales margins and number of customers. 24

Figure 3.3 Differences in cost and pricing. 26

Figure 3.4 the producers of open source products 33

Figure 3.5: The Distributor Business Model..... 36

Figure 3.6: The Software Producer (Non-GPL Model) Business Model..... 37

Figure 3.7: The Software Producer (GPL Model) Business Model..... 38

Figure 3.8: Third Party Service provider Business Model..... 40

List of tables

Table 1 Country Distribution of Firms	45
Table 2 Distribution of the sample firms By Year of Foundation and Size	46
Table 3 Explanatory variables used in the estimation of the econometric model.	50
Table 4 Descriptive statistics of explanatory variables of econometric models.....	51
Table 5 Correlation matrix of the explanatory variables included in the model.	52
Table 6 DMainlyOSS performance over Dmainlyprop in terms of lnheadcount.....	52
Table 7 DMainlyOSS performance over Dmainlyprop in terms of lnsales.....	53
Table 8 DMainlyOSS performance over Dmainlyprop in terms of lntotalassets.....	53
Table 9 MOSS performance in terms of lnheadcount.	54
Table 10 MOSS performance in terms of lnsales.	54
Table 11 MOSS performance in terms of lntotalassets.	54

1 Introduction

In recent years, there has been a surge of interest in open source software development. Interest in this process, which involves software developers at many different locations and organizations sharing a code to develop and refine software programs, has been stimulated by three factors:

- The rapid diffusion of open source software. A number of open source products, such as the apache web server, dominant their product categories. In the personal computer operating system market, International Data Corporation estimate that the open source program Linux has between forty to fifty million users worldwide, with a 200% annual growth rate. Many observers believe it represent a leading potential challenge to Microsoft Windows in this important market segment.
- The significant capital investments in open source projects. Over the past few years, numerous major corporations, including Hewlett Packard, IBM and Sun have launched project to develop and use open source software. Meanwhile, a number of companies specializing in commercializing Linux, such as Red Hat and VA Linux, have completed initial public offering and other open source companies such as Cobalt Networks, Collab.Net, Scriptics and Sendmail have received venture capital financing.
- The new Organization structure. The collaborative nature of open source software development has been hailed in the business and technical press as an important organization innovation.

Open-source software, sometimes referred to as Free Software, (since they do not always refer to exactly the same concept) is software whose source code is made available to any and all users of that software. It differs from proprietary software in that the users of a proprietary software program only have access to the binary executable files. Well-known examples of OS software include the Linux operating system, the Apache web server, the Perl programming language, Mozilla's Firefox Internet browser and so on.

At first glance, the behavior of programmers who choose to write code for free, open-source projects runs counter to traditional economic theory [1]. The open source development process converts a normally proprietary product, i.e. software, into a public good, developed using a voluntary-contribution mechanism. However, not only does OS/FS software exist, but it has been embraced, in varying degrees, by a number of high-profile IT firms including IBM, HP, Cisco, and Apple.

Yet to an economist, the behavior of individual programmers and commercial companies engaged in open source processes is startling.

1.1 Motivation

The advent of open source software has produced more than lower software costs for users. It has also caused major changes in the economic interaction among players in the software ecosystem. For many, open source embodies a specific approach to software development—even a lifestyle. But it's also sound business strategy. Ron Goldman and Richard Gabriel suggest that companies should use open source software to grow their user communities and build an ecosystem around their products and services [2].

Open source software is typically free and comes with the source code needed to adapt it to users' needs. Most open source licenses let users redistribute the software, including possible changes, and charge for redistribution as long as source code changes are publicly available (www.opensource.org).

1.2 Goal and Objective

Software companies participating in the OSS development projects tend to offer their own software with more restrictive licenses than other software companies. The empirical research further hints that software firms do not, however, try to coordinate the further development of their own software by using the GPL but rather aim at responding to their customers' needs by offering more flexible licensing terms[3].

This paper seeks to make a preliminary exploration of the economics of open source software. And find out the impact of business strategies over open-source firm's performance.

2 History of Open-Source Software

2.1 The Analytic Problem of Open Source

Coca-Cola sells bottles of soda to consumers [4]. Consumers drink the soda (or use it in any other way they like). Some consumers, out of morbid curiosity, may read the list of ingredients on the bottle. But that list of ingredients is generic. Coca-Cola has a proprietary 'formula' that it does not and will not release. The formula is the knowledge that makes it possible for Coke to combine sugar, water, and a few other readily available ingredients in particular proportions and produce something of great value. The bubbly stuff in your glass cannot be reverse-engineered into its constituent parts. We can buy it and we can drink it, but we can't *understand* it in a way that would empower us to reproduce it or improve upon it and distribute our improved cola drink to the rest of the world.

The economics of intellectual property rights provides a straightforward rationalization of why the Coca-Cola production 'regime' is organized in this way. The problem of intellectual property rights is about creating incentives for innovators. Patents, copyrights, licensing schemes and other means of 'protecting' knowledge assure that economic rents are created and that some proportion of those rents can be appropriated by the innovator. If that were not the case, a new and improved formula would immediately be available for free to anyone who chose to look at it. The person who invented that formula would have no claim on the knowledge or any part of the profits that might be made from selling drinks engineered from it. The system unravels, because that person no longer has any 'rational' incentive to innovate in the first place.

The production of computer software has typically been organized under a similar regime. You can buy Microsoft Windows and you can use it on your computer but you cannot reproduce it, modify it, improve it, and redistribute your own version to others. Copyright provides legal protections to these strictures, but there is an even more fundamental mechanism that stops you from doing this. Just as Coca-Cola does not release its formula, most software developers do not

release their *source code*, the list of instructions in a programming language that comprise the recipe for the software. Source code is the essence of proprietary software. It is a trade secret. Proprietary source code is the fundamental reason why Microsoft can sell Windows for a non-zero price, and distribute some piece of the rents to the programmers who write the code -- and thus provide incentives for them to innovate.

Open Source software inverts this logic. The essence of open source software is that source code is 'free' -- that is -- open, public, non-proprietary. Open Source software is distributed with its source code. The Open Source Definition (which I discuss in greater detail later) has three essential features:

- It allows free re-distribution of the software without royalties or licensing fees to the author
- It requires that source code be distributed with the software or otherwise made available for no more than the cost of distribution
- It allows anyone to modify the software or derive other software from it, and to redistribute the modified software under the same terms.

There exist several hundred or perhaps thousands of open source 'projects', ranging from small utilities and device drivers to Sendmail (an e-mail transfer program that almost completely dominates its market) to WWW servers (Apache) and a full operating system -- Linux. These projects are driven forward by contributions of hundreds, sometimes thousands of developers, who work from around the world in a seemingly unorganized fashion, without direct pay or compensation for their contributions.

Open Source can be characterized variously as:

- A particular methodology for research and development
- The core of a new business model (free distribution of software means that new mechanism for compensation and profit need to be created)
- The social essence of a community, a defining nexus that binds together a group of people to create a common good.
- A new 'production structure' that is somehow unique or special to a 'knowledge economy' and will transcend or replace production structures of the industrial era.
- A political movement.

2.2 Definition of Open Source Software

A computer program is a series of instructions of tasks to performs given to a piece of hardware. There are a number of different “Languages” roughly analogous to human languages, which can be used to write a computer program. However, the only instructions which hardware can understand are a series of 1’s and 0’s, representing the presence or absence of an electrical current in a circuit. Computer programming languages – for examples , C, C++, Java, BASIC or Perl etc exist to provide an interface between the “machine language” of 1’s and 0’s, which is extremely tedious for the human mind to work with, and human though about the goals of a program. Specialized programs, knows as “compilers,” can translate programs written in a higher-level programming language into the machine language.

Therefore, when a company or individual distributes a computer program, they can make a choice about whether to distribute only the complied code, which appears as 1’s and 0’s and from which it is virtually impossible to recapture the original program, or they can distribute the higher-level source code, which carriers much more information about the structure and purpose of the program. The most basic definition of open-source software is a software program for which the source code is available. The price of the software is irrelevant to this definition, although it is often the case, because of the culture of computer programmers and some others aspects of the software world, that open-source software is available for free over the internet. However, closed-source software can also be distributed for free; then it is known as “shareware” and usually used to build demand for a later or more sophisticated version. And open-source software is not necessarily free; its copyright license can put a number of specifications on its use including payment. The theoretical differences between open and closed-source software come from their different developmental and distributional models.

Open source software is a particular kind of public good, characterized by non-excludability I mean once software has been put on the internet, it can be distributed quickly and freely and perfect jointness of supply.

2.3 A brief history of open source software:

Computer Science is a relatively new discipline, and it was born in government and university programs that supported an open culture of sharing code and collaborating.

Therefore, the idea of “open source software” is as old as computer programming itself, and can be compared to the idea of peer review of academic articles. The history of computer science and the interaction of the public and private sectors can be seen as the story of investors trying to balance the tremendous growth that sprung from encouraging open collaboration, while still trying to capture the value that resulted.

The “ARPANET”, the prototype of the modern internet, was built by the Defense Department in 1969 and linked hundreds of universities, defense contractors and research laboratories [5]. It gave an additional boost to the sharing of code by computer scientists all across the country, and it wasn't long before early versions sprung up of the standard bulletin boards and email lists used by open source developers today. Some important centers of computer science research included MIT's Artificial Intelligence lab, Stanford University's Artificial Intelligence Laboratory (SAIL) and Carnegie-Mellon University's computer science department. ARPANET helped create a critical mass of information by allowing an easy exchange of ideas between people from all these different places.

2.3.1 The University of Berkeley and the development of UNIX

UNIX, the first advanced and important computer operating system, arose as a result of a characteristic collaboration between the semi-public (Berkeley's computer science department) and private (Bell Labs.) Ken Thompson, a Bell employee, modified the time-sharing operating system, and realized that by writing an operating system in the C programming language instead of high-level machine language, it could be much more portable, flexible and understandable. The first implementation of UNIX came in 1969- the year ARPANET was created- and after a decade of work UNIX had been successfully “ported” to several different hardware systems. If UNIX could present the same face, the same capabilities, on machines of many different types, it could serve as a common software environment for all of them. No longer would user have to

pay for complete new designs of software every time a machine went obsolete. Hackers could carry around software toolkit between machines, rather than having to re-invent the equivalent of fire and the wheel every time.

The history of software development can be usefully viewed in terms of the framework of the clash between a collegial, open model and a corporate, closed model. This clash is exemplified by the problems that sprung up in the relation between AT&T and Berkeley. Their collaboration started breaking down when a group of computer scientist at Berkeley developed a distribution of UNIX called BSD (Berkeley Software Distribution).

However, their code used part of the source code that had originated from AT&T, and the company sued Berkeley to prevent them from offering a free competition to its operating system, which earned such lucrative licensing fees.

Meanwhile, in the late 1980s, the computer (PC) market was beginning to heat up, and a company called Microsoft was, through collaboration with IBM, able to seize a dominant share of the operating system market for these domestically used machines. The increase in PC use and the explosion of the use of the Internet by private consumers was joined by a flowering of other companies that kept their code private and earned a profit, such as Netscape, the creator of the most widely used browser for surfing the internet.

However, open source development never stopped during this time. Because the Berkeley BSD distribution was tied up in the lawsuit by AT&T, there rose a huge demand for an open-source operating system of an equal power and sophistication to UNIX. One of the most important projects was Linux, Started by Linus Torvaldis, who wrote a kernel (based in large part on earlier work done by Richard Stallman of the Free Software Foundation).

2.3.2 The birth of Linux

As befits any institution with a passionate following, the story of Linux's creation has become hacker legend. It started with a Finnish graduate student, Linus Torvaldis, who became frustrated with the UNIX operating system in 1991, and revamped it into the Linux core program. However, to say that Linus "created" Linux would be to misrepresent his most important accomplishment, which was the perfection of the open-source system of software development.

As Andrew Leonard puts it in his book-in-progress on the history of open-source software, “Torvaldis exploited the Net’s facility for bringing people together...using e-mail and Usenet, he nurtured a worldwide community of freely collaboration programmers”[6]The network of programmers, all of whom used Linux, suggested bugs that needed to be fixed, and submitted code that fixed those bugs. Programmers had used the open-source system ever since the first days of the Internet, but Linux is the biggest, most complex, and arguable the most successful open-source project ever to exist.

Linus continued to work programming Linux, but he also acted as a “benevolent dictator”: he received submissions from anyone else using the operating system, chose the best ones, and periodically, sometimes even daily issued new releases of Linux’s source code, crediting the contributors. By 1994, Linux version 1.0 was ready for general distribution. The operating system was steadily improved, becoming more stable and more complete, with every new version numbered in such a way that potential user could make a choice either to run the last version designated “stable” or risk bugs to get new features.

The Linux core handles all the basic functions of an operating system and is still being managed by Linus. Meanwhile, there is an expanding galaxy of Linux-compatible applications, all of them “copylefted,” which are available from different places on the internet .Linux packages, available on CD-ROM, with the Linux kernel and a coherent selection of accessories, some of which may be developed by the Linux Company itself, others of which are simply reproduced according to their “copy lefts.” Last few years were an intense excitement in the investment community about Linux. It’s now available for download from many sites, including <http://www.linuxtoday.com/>, on CD-ROM packages from various companies such as Red Hat Linux, and even comes bundled with some Dell PCs. Linux poses a direct, short-term revenue and platform threat to Microsoft, particularly in server space- Linux is making a progressively more credible argument that OSS software is at least as robust-- if not more—than commercial alternative.

2.3.3 The growth of the Apache web server group

The Apache web server group is another open source project that started in 1995, when the most used server software on the web was a public domain HTTP daemon developed by Rob McCool at the National center for Supercomputing Applications. A group of webmasters, who had developed their own private extensions and bug fixes, gathered together to join and coordinate these changes, something that helped them immensely with all of their individual jobs. The core group of developers, including Brian Behlendorf (who works for O’reilly Software Associates), and ken Coar (who works for IBM), formed the Apache Software Foundation, a non-profit corporation, in July 1999[7]. Their goal was to provide a legal framework for Apache’s development, but members of the Apache community remain solidly pragmatic and focused on their regular jobs. The Apache web server group has a completely different culture and goals from the Linux group. Their motivation for collaborating in an open source environment is much more closely tied to the reduction of development costs in their existing work, while a contributor to Linux is more likely to be motivated by “signaling incentives.” Today, web servers using the Apache software account for about 65% of the total

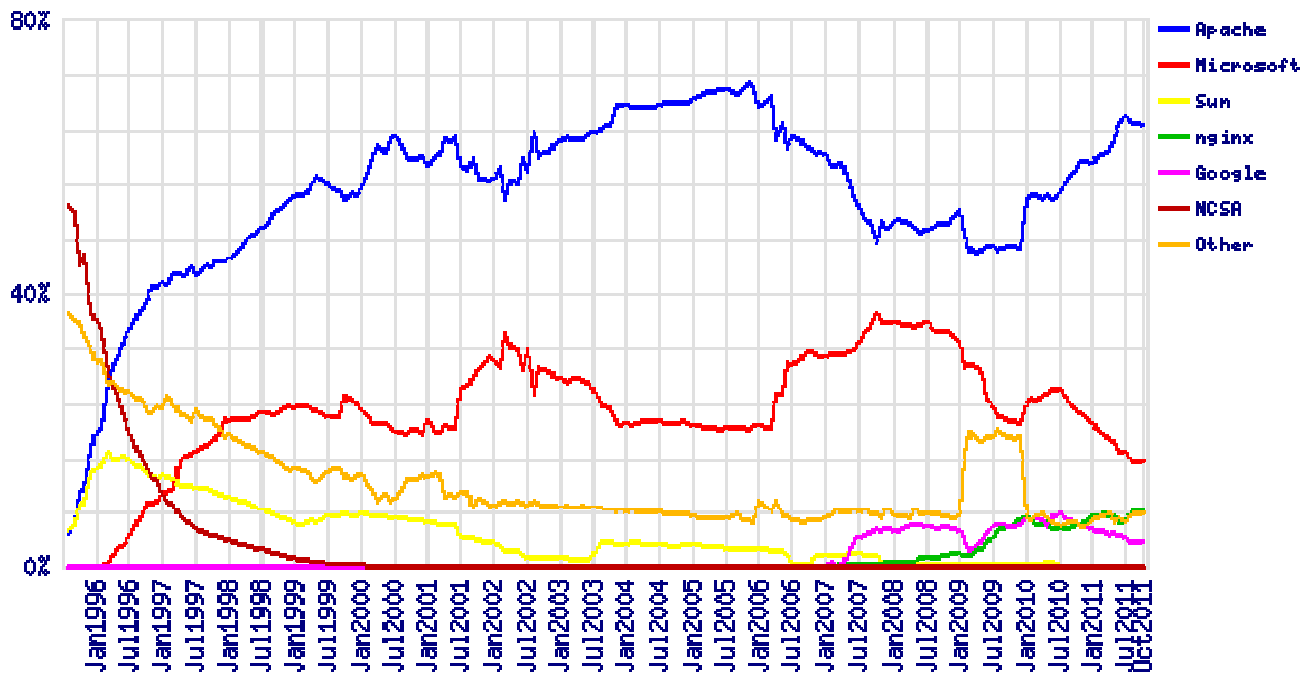


Figure 2.1: Market Share for Top Servers Across All Domains (www.netcraft.com)

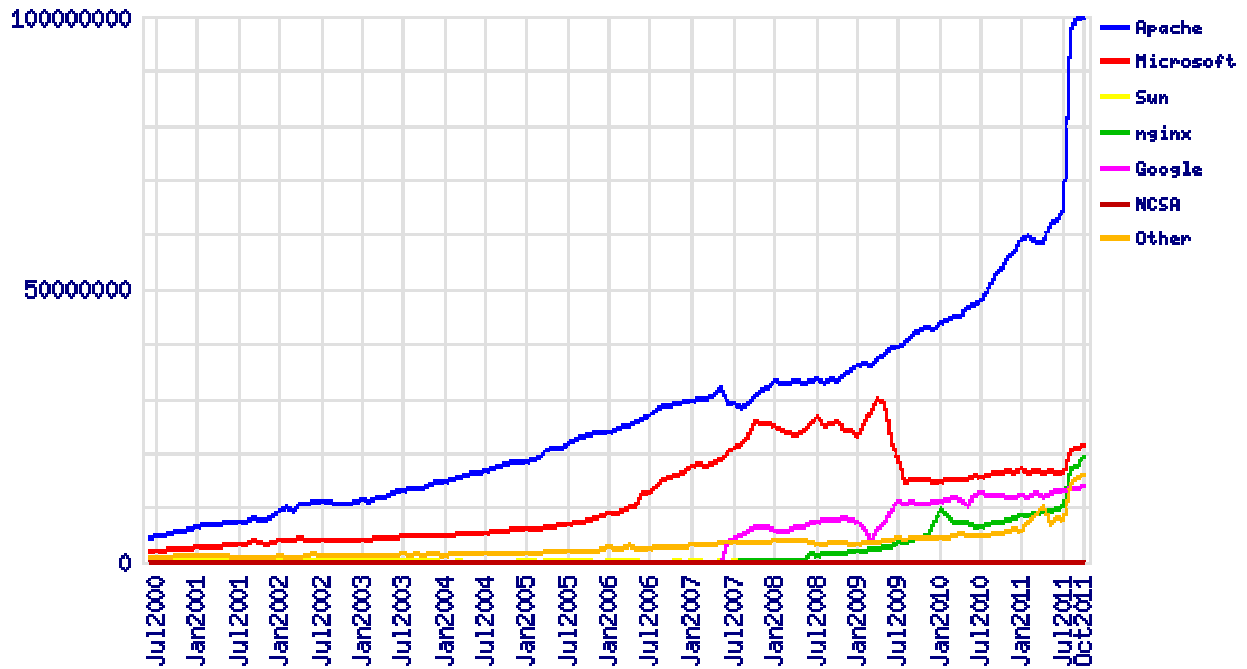


Figure 2.2: Totals for Active Sites Across All Domains June 2000 - October 2011(www.netcraft.com)

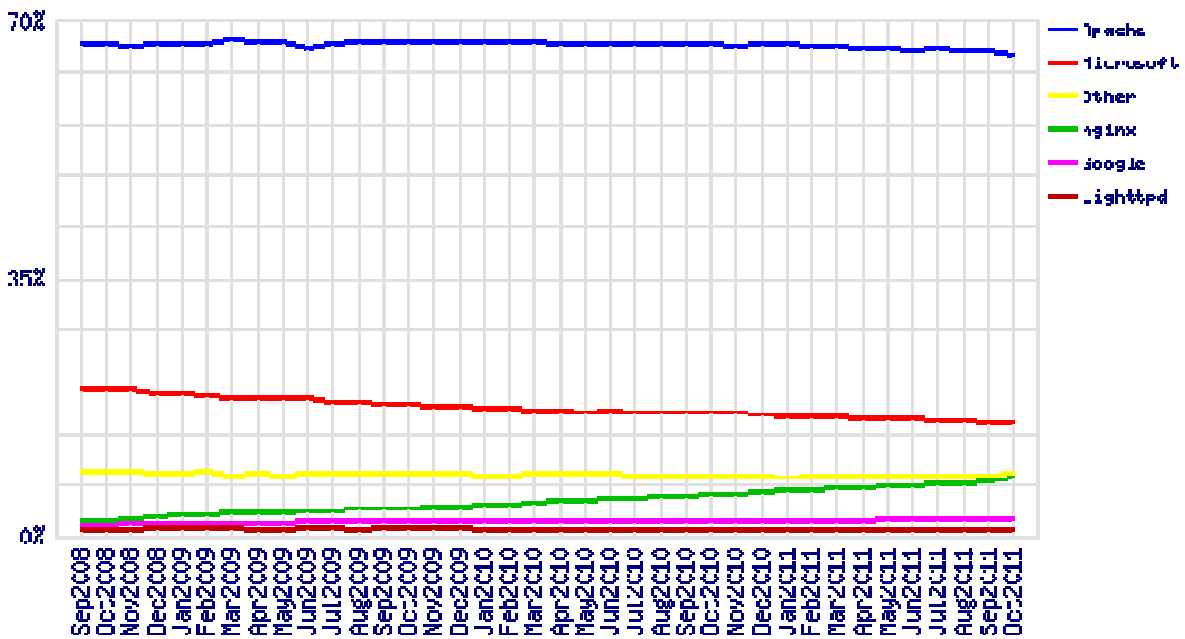


Figure 2.3: Market Share for Top Servers Across the Million Busiest Sites September 2008 - October 2011(www.netcraft.com)

Market (source: www.netcraft.com). Every eight seconds another Apache-based websites joins in the existing 10 million on the web. Apache web servers are well known for being stable and secure.

2.3.4 Netscape- Rise, fall and possible recovery?

Netscape, which started as one of the early, trend setting closed-source software companies, decided in 1999 to go open-source. Why did they change their mind? It was originally very economically an advantageous for Netscape to keep control of their source code. They had a “killer app,” code which they had developed which did not exist anywhere else, and the market for talented programmers’ familiars with the then-nascent internet was illiquid enough that it would be extremely hard for other companies to duplicate their efforts.

Moreover, the most important target users of Netscape were relatively computer-inexperienced users both in businesses and households. This meant that there was a fairly large training cost involved in beginning to use the Netscape browser. A company that trained all its employees to use Netscape would be unlikely to shell out even more money to train them again to use a different browser, and consumers would similarly be reluctant to learn again. Therefore, there was a large amount of profit for Netscape to capture by keeping control of its source code. However, over the course of the 1990s, Microsoft developed a competing Internet browser, Internet Explorer (IE) and began to use their monopolistic distribution channels to force IE’s dominance over Netscape. By the late 1990s Netscape had lost a considerable amount of market share, and Marc Andreesson, the CEO of Netscape, decided to release Netscape’s source code and attract a group of open source programmers in an attempt to find a new way to challenge Microsoft. So far, this attempt has been a relative failure, especially compared with the success of Linux and the Apache server, in attracting both developer and users. The Mozilla browser is an interesting case study to contrast to the success of Linux and especially Apache, to illuminate the market conditions and developmental choices that lead to incentives for programmers to volunteer their time to an open source project, and for consumer to prefer an open source product.

2.4 OSS Literatures

Open source software (OSS) is changing the way organizations develop, acquire, use and commercialize software.

Eric Raymond describes the development of OSS as a bazaar-like activity driven by volunteers, and claims that OSS is cheaper, has fewer defects, gets improvements faster, and is generally better than "other kinds" of software [8, 9]. Based on the Apache and the Mozilla projects, Mockus et al. [10] describe OSS development as controlled by groups of core developers and supported by large communities of contributors. They hypothesize that the OSS products have lower defect density than commercial software and that OSS development rapidly responds to user requests. Others, e.g. Crowston and Howison [11] and Scacchi [12], support this view and claim that the development in OSS communities is distinctly different from traditional software development.

This view of OSS and OSS development as being something radically different has triggered research on a variety of topics. These include OSS seen as a new innovation model [13], the motivations of OSS developers [14], OSS business models [15] and a wide spectrum of other research topics in computer science, management and organization science, social science, psychology, economics, and law [16].

Software engineering research has for instance studied self-organizing in OSS communities [17, 18], user-to-user support [19], knowledge management [20], and quality assurance [21]. Software engineering researchers have additionally used OSS products to study general software engineering problems like evolution [22], cloning [23], and the use of metrics to identify error prone classes [24].

However, the view that the development of OSS is something radically different from traditional software development is questioned by, for instance Fitzgerald [25] and Fugetta [26]. Østerlie and Jaccheri [27] offer a critique of how OSS development has been described as a homogeneous phenomenon in the software engineering research literature. The literature has not reflected

the variety observed in the OSS phenomenon, but rather has focused on large, successful, and community-driven OSS projects. Moreover, Capiluppi [28] provide evidence that the majority of OSS projects struggle to attract contributors, Noll [29] shows that OSS can also be developed inside commercial software development companies without any active communities, and Stamelos [30] show that the quality of OSS software is not always as good as expected. Finally, Fitzgerald [31] argues that the OSS phenomenon has evolved into a more commercially viable form where volunteers and commercial organizations collaboratively contribute to evolving the phenomenon.

There are thus conflicting views on what the OSS phenomenon actually is and there is not even consensus on which label to use on the phenomenon. We acknowledge that there are (minor) differences between open source, free software, and free (libre) open source software (FOSS/FLOSS). However, this ongoing debate is beyond the scope of this paper.

We will in particular look at three aspects of it related to organizations:

(1) The use of software products licensed with a license approved by the Open Source Initiative [32], (2) the interaction with the communities surrounding many OSS products, and (3) the use of the collaborative software development practices often associated with many of these communities.

Summary

Open source software is a particular kind of public good, characterized by non-excludability I mean once software has been put on the internet, it can be distributed quickly and freely and perfect jointness of supply.

Collaborative Open Source software projects such as Linux and Apache have demonstrated, empirically, that a large, complex system of code can be built, maintained, developed, and extended in a nonproprietary setting where many developers work in a highly parallel, relatively unstructured way and without direct monetary compensation.

3 Economics of open-source software

3.1 The Economic Motivation of Open Source Software

Open source software has changed the rules of the game, impacting significantly the economic behavior of stakeholders in the software ecosystem. In this new environment, developers strive to be committers, vendors feel pressure to produce open source products, and system integrators anticipate boosting profits.

The advent of open source software has produced more than lower software costs for users. It has also created major changes in the economic interaction among players in the software ecosystem.

For many, open source embodies a specific approach to software development---even a lifestyle. But it's also sound business strategy. Ron Goldman and Richard Gabriel suggest that companies should use open source software to grow their user communities and build an ecosystem around their products and services [33].

Open source software is typically free and comes with the source code needed to adapt it to users' needs. Most open source licenses let users redistribute the software, including possible changes, and charge for redistribution as long as source code changes are publicly available (www.opensource.org).

There are two types of open source software [34]. *Community open source* is software that a community develops. Rather than a single corporate entity owning the software, a sometimes broad community of volunteers determines which contributions are accepted into the source code base and where the software is headed. Individual developers, the *committers*, and not a specific company, make decisions about the software, as in the case of the Apache Web server (<http://httpd.apache.org>)

Commercial open source is software that a for-profit entity owns and develops. The company maintains the copyright and determines what is accepted into the software code base and what to implement next, as in the case of MySQL and its MySQL database (www.mysql.com).

Prior work on community open source economics focused mostly on labor economics, that is, the frequently surprising amount of volunteer work that goes into open source software. Eric Raymond notes that developers contribute to open source projects for the personal gratification that comes from increasing their reputation among peers [35].

In the following, we will discuss the economic motivation of system integrators, independent software vendors, and software developers to explain these seeming contradictions.

3.2 SYSTEM INTEGRATOR PERSPECTIVE

Large system integrators, or solution providers, stand to gain the most from open source software because they increase profits through direct cost savings and the ability to reach more customers through improved pricing flexibility. Every dollar a system integrator saves on license costs paid to a software firm is a dollar gained that the customer might spend on services.

3.2.1 IT solutions demand curve

Customers typically want information technology providers to deliver "solutions." A solution solves a customer's IT problem, freeing the customer to focus on business rather than IT. A comprehensive solution comprises hardware, software, and services. Indeed, the IT industry earns its living by removing or reducing customers' IT worries.

System integrators deliver solutions by selling a stack of hardware, software, and services as one product. That allows the customer to talk to one company, rather than many. Figure 3.1a

illustrates this stack together with the customer demand curve.

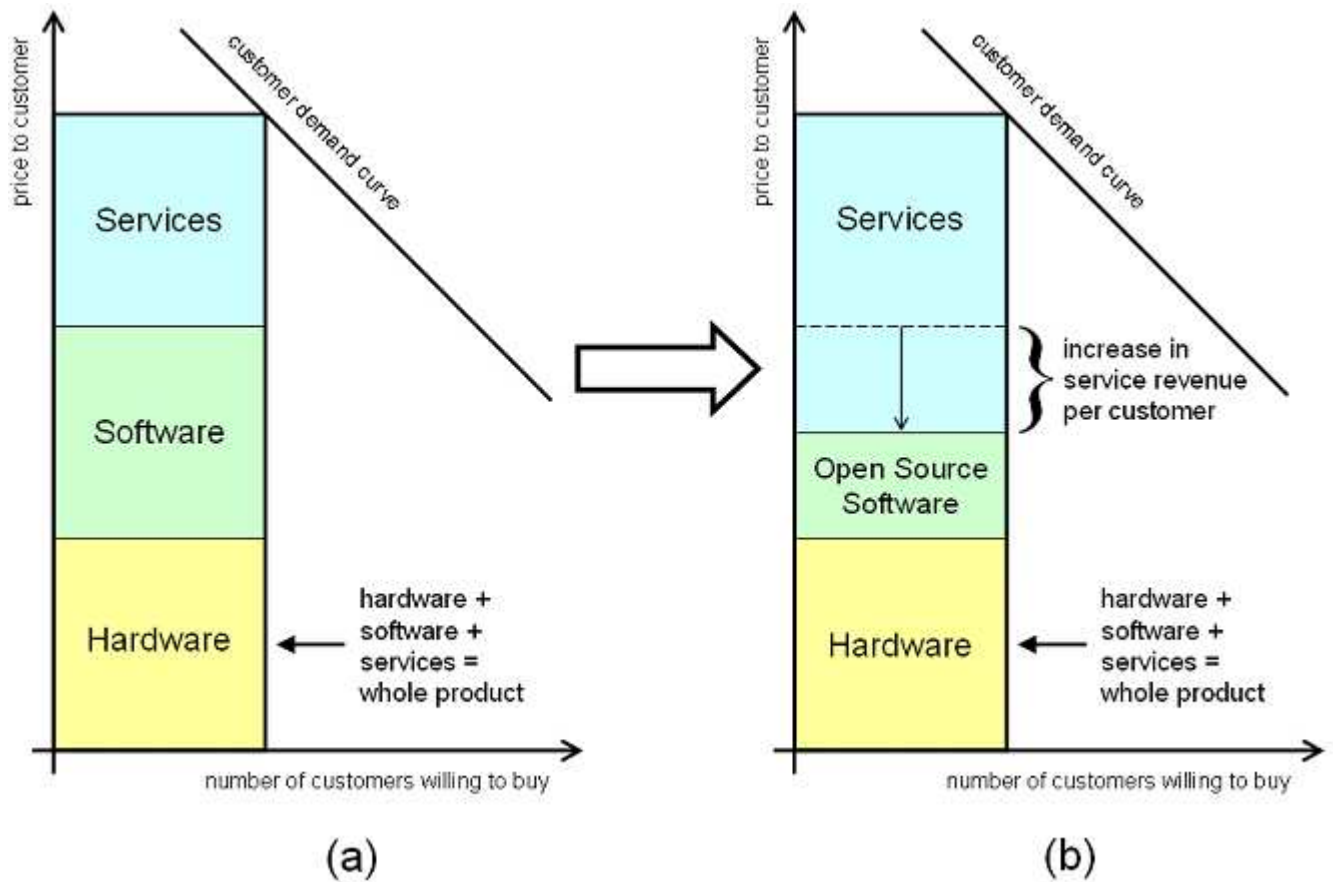


Figure 3.1. IT solutions demand curve.

Figure 3.1. IT solutions demand curve. (a) System integrators sell a stack of hardware, software, and services. (b) Integrators can charge customers similar prices even if they use open source software.

The demand curve shows how many customers are willing to buy the system integrator's solution at a given price. On the Y axis is the customer's cost to purchase a solution, and on the X axis is the number of customers who are willing to pay for that solution at the given price. The form of the demand curve varies depending on what is being sold. However, in general, the demand curve is downward sloping: The lower the price, the more customers are willing to buy.

A system integrator's profits depend on which of the stack's components it owns and which it must buy. Usually, a system integrator's stronghold is services, which puts together the hardware and software pieces to meet the customer's need. However, if the system integrator owns only the

services component, it will have to pay other companies for the software and the hardware and thereby share revenue, leaving less profit for itself.

It's therefore in a system integrator's interest to acquire hardware and software as cheaply as possible. Open source software, if an option, is typically much cheaper than closed source software; hence its use increases profits for the system integrator.

Figure 3.1b illustrates how with stable supply and demand, more money is made in the services part of the value stack if software cost goes down.

Software cost savings aren't easily passed on to customers for two reasons: First, customers tend to care about the whole product rather than individual components; second, large system-integration projects are complex and new competition doesn't spring up easily. Thus, system integrators can maintain their prices.

While this is one good reason for system integrators to support open source software, there's another equally compelling reason for them to support and contribute to open source software.

3.2.2 Business growth

The simple value stack that Figure 3.1 illustrates suggests that system integrators charge customers only one price. In reality, the system integrator can charge varying prices for a total solution to a prospective customer's problem. Only one thing is certain: A system integrator will want to at least cover its costs.

The price charged per customer that Figure 3.1 shows can be split into the system integrator's service cost, plus the markup or margin needed to make a profit. If the system integrator owns just the services part of the stack, the cost for providing that service defines the lower price limit for the work. In a reasonably competitive market, the system integrator will accept deals above this limit if it has the resources.

This limit, together with the demand curve, determines the maximum number of customers the system integrator can sell to and take on, as Figure 3.2a illustrates.

Switching from more expensive closed source software to less expensive open source software increases the profits of a sale through the money saved on the software. It also reduces the lower price limit for possible deals and puts a new set of more price-sensitive customers within reach. Not only does open source software improve profits on the original individual sales, it also increases the total number of potential customers.

Figure 3.2b shows how a switch from closed source to open source software results in more potential customers. And more potential—and presumably satisfied—customers mean higher sales and profits. The total profit is represented as the area of the gray triangle under the demand curve, showing the increase in profits when moving down the curve. Since in reality a system integrator might own many of a total solution's components, including software and hardware, more customers mean more profits through these components as well.

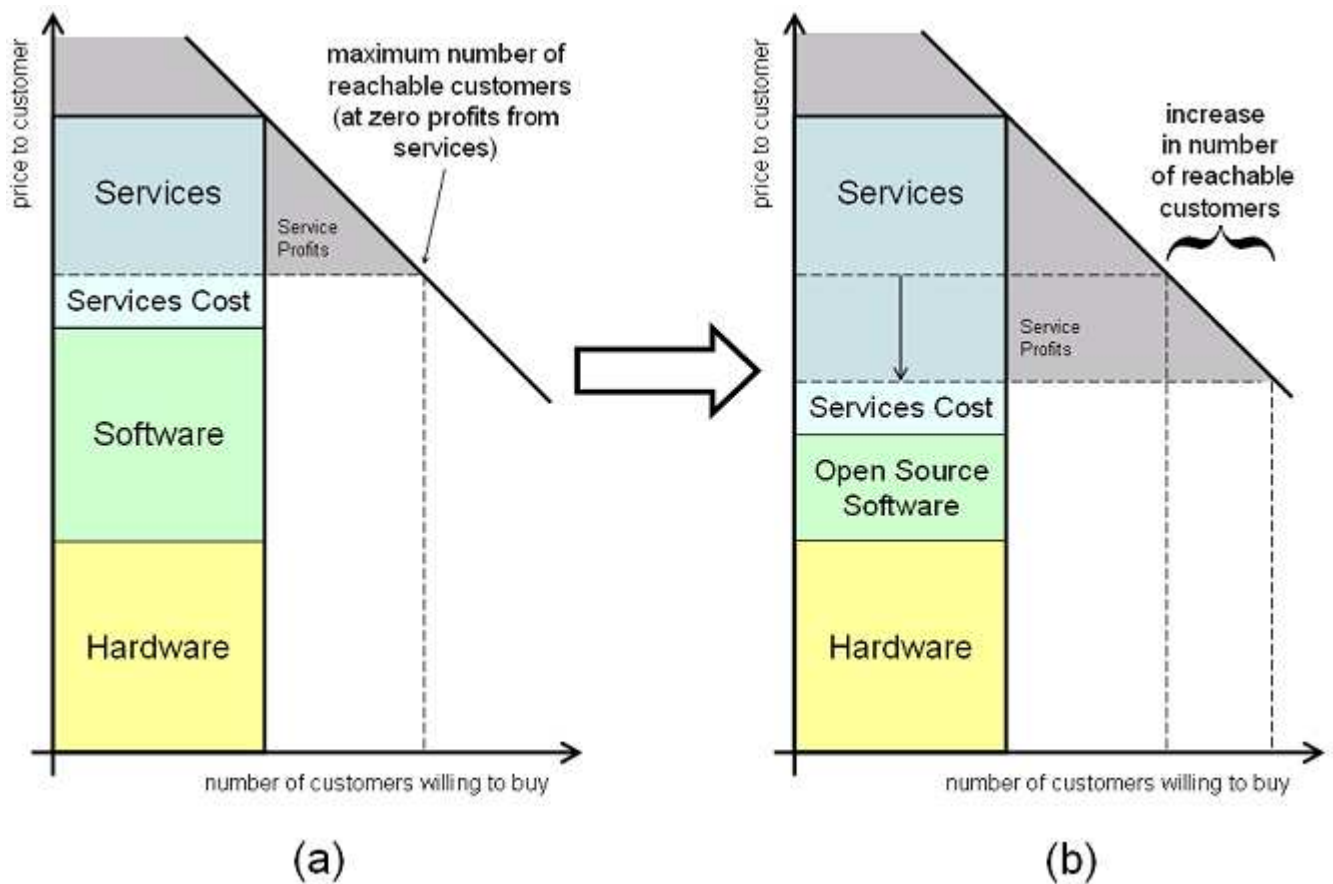


Figure 3.2 Sales margins and number of customers.

Figure 3.2 Sales margins and number of customers. (a) The lower price limit determines the customers the system integrator takes on. (b) Switching from closed source software to open source software can result in more customers and higher profits.

3.2.3 Pressures in the IT value stack

If it were up to the system integrators, all software would be free (unless they had a major stake in a particular component). Then, all software license revenue would become services revenue. To this end, I believe that system integrators prefer community open source over commercial open source. Only community open source software prevents vendor lock-in.

Community open source ensures that prices for software support are subject to market forces rather than one owning corporation. Community open source is a strategic weapon for system integrators to squeeze out proprietary as well as commercial open source software vendors.

3.3 SOFTWARE VENDOR PERSPECTIVE

Independent software vendors provide only a few software products, sometimes just specializing in one. Understanding the independent software firm strategy requires comparing open source software and closed source software cost and pricing.

3.3.1 Software cost and pricing

Why is open source software typically much cheaper than closed source software? In a closed source business, most of the investment in new software comes from shipping the first copy. The initial investment is recouped with increasing sales. The additional cost of producing and selling another copy is small, consisting of producing another CD or allowing for another download plus providing the (frequently minimal) free support that comes with a software license.

As the number of copies sold increases, the average cost per copy declines and the profits rise. Figure 3.3a illustrates the long-run average cost curve for a single software product.

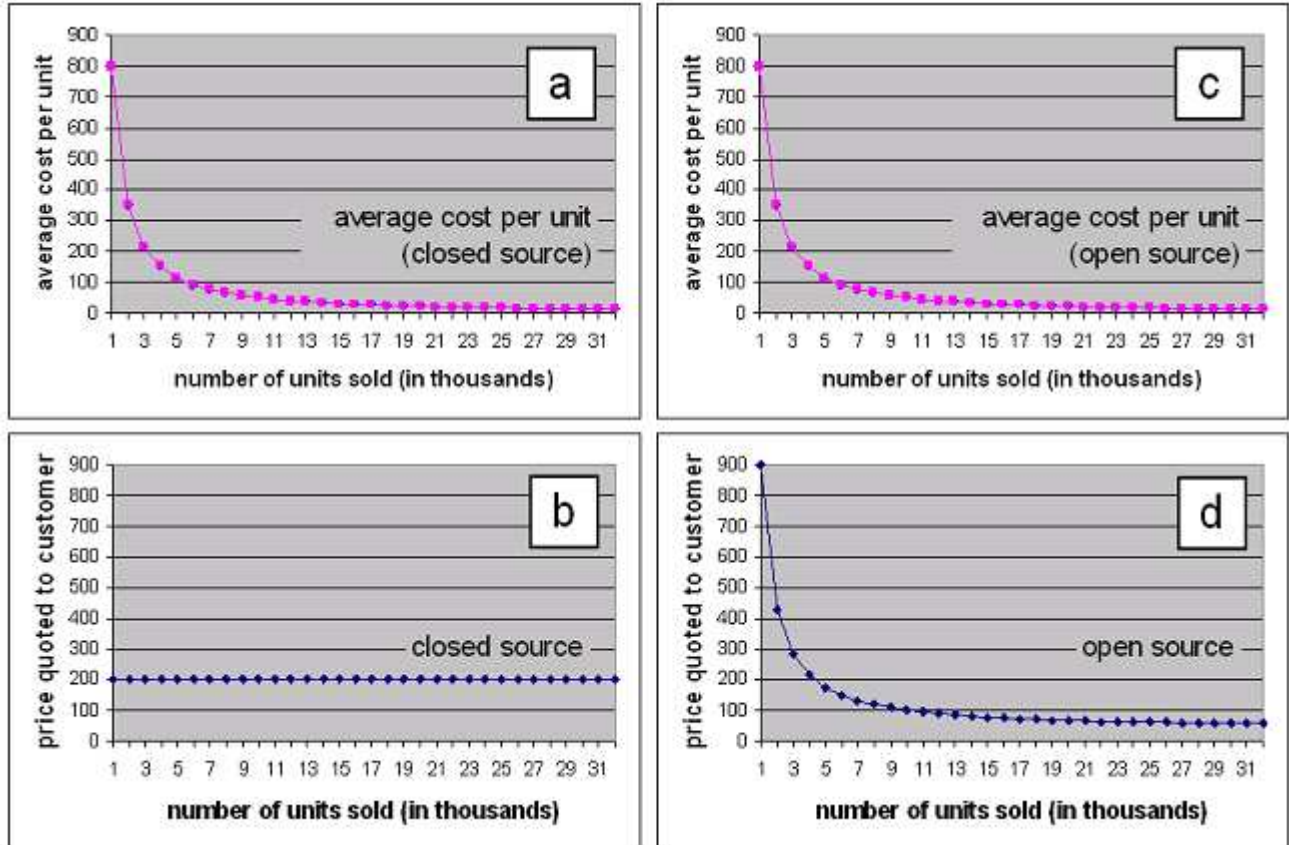


Figure 3.3 Differences in cost and pricing.

Figure 3.3 Differences in cost and pricing. (a) and (c) show a similar average cost per unit for closed source and open source software, while (b) and (d) indicate the relationship between price and number of units sold for open and closed software.

The more mature a market for a specific software component, the higher the investment in the existing products, the higher the barrier to entering this market, the more established the existing players, and the more stable the price for the software component. A common scenario is "the 800-pound gorilla" firm that dominates a market and is surrounded by smaller players catering to market niches [36].

In such a market, the leading software vendor sets a price that maximizes its profits. Since the market is fairly transparent, the vendor can set just one pricing schedule, offering the product to different customers at the same price. (This is in contrast to the highly individual deals of large system integrators.) The result is frequently a flat price, as Figure 3.3b shows. Remarkably, the

price of the proprietary closed source software doesn't directly depend on the actual cost incurred to develop, maintain, and provide the software.

The profit-maximizing price is largely independent of cost; the cost provides only a lower limit. Competition that drives prices closer to cost can't spring up easily due to the large initial investment a software product requires.

In a community open source situation, no such market-entry barriers exist. Given the right license, anyone can set up a company and start selling software. What the company will sell, of course, isn't the software itself, but its provision, maintenance, and support.

Because anyone can enter an attractive open source market, competition is fierce, and pricing will be based on markup over cost. If the markup is too high, new companies will enter the market; if it's too low, companies will leave the market. Moreover, the more mature the product, the lower the overall price.

Figure 3.3c shows the total cost of developing open source software. The total cost and the resulting average cost per copy sold is mostly the same as for the closed source solution. The main difference, of course, is that the different contributing companies now share this cost.

Figure 3.3d illustrates the pricing of open source software from a single firm's perspective. Because of the competition, the price charged for providing the software plus support is based on markup over cost.

Different firms will have different costs depending on their share of contributions to the open source project. However, with increasing project share, the company can charge higher prices because customers are likely to receive better service. The basic relationship remains unaffected: Price is markup over cost and varies depending on cost.

Customers love this situation because prices are substantially lower than in the proprietary situation. System integrators love the situation even more because they can squeeze out proprietary closed source software.

3.3.2 Generating software profits

A system integrator can increase its profits if it reduces the software cost. By reducing software cost, it can move down the demand curve and sell to more customers.

Proprietary software is the main obstacle to doing this: It cuts into profits on an individual sale and reduces overall pricing flexibility. Hence, system integrators have a high interest in turning proprietary software markets into software markets with at least one viable community open source product.

Before the advent of open source software, entering an established and well-defended market was a risky proposition. With increasingly well-understood open source processes, setting up an open source project competing with an established proprietary market leader's product is much less risky and carries a significantly higher chance of success than before. But it's not just a specific system integrator that will want to do that. It's pretty much everyone who isn't the proprietary source market leader.

My best option now is to open source your product. You might be reducing the market's overall return on investment, but at least you'll have a second chance at satisfying your own investors by making your company a successful open source business. You'll be in good company. With a proper license for your open source product, you might well receive help from the system integrators, customers, and software vendors higher in the IT stack.

3.3.3 Commercial Open Source

With such gloomy prospects for proprietary source businesses, independent software vendors have sought business models for harnessing open source software's benefits while gathering some of the profits of a closed source business. The answer is commercial open source.

The key differentiator between community open source and commercial open source is whether a community or a single entity like a corporation holds the power to make decisions about the project.

Commercial open source software is typically available for free to nonprofit users. Sometimes commercial use is free as well. Usually, companies make money by providing support services. Sometimes they make additional money by selling proprietary software enhancements.

Like community open source, commercial open source is available in source code form. Unlike community open source, however, one company controls commercial open source. This way, commercial open source software can gather some of the benefits of community open source: faster adoption, free and speedy user feedback, and possibly volunteers' code contributions. This approach is mostly a marketing strategy, however, because the company that owns the software still must do the development. Hence, the company must employ and pay the software's developers.

During the early days of an open source project, this is an advantage, as the company can provide clear direction and gather together more resources than community open source projects typically can. As the project matures, this can turn into a disadvantage, as a competing community open source project might have more resources at hand in the form of volunteers.

3.3.4 Open source service firms

If it isn't possible to be a profitable proprietary source business, what does it mean to be a successful open source business? The market's answer is the open source service company, which comes in at least two kinds [37]. One provides first-level support and implementation services; the other provides second-level support, training, and development services.

Clients of the first kind of firm are typically IT users who employ the firm's services to put the open source product into place in their IT operations. Clients of the second kind of firm typically need to get trained on the product or need to have a technical problem fixed that they can't handle themselves.

The strength of a service business usually lies in its ability to:

- recruit and retain the right people
- reliably set up and execute specific service processes, and
- Bring to bear expert domain knowledge and unique intellectual property.

In the open source situation, this is usually labor economics. Technical skills around the open source product are a key part of determining an employee's value to a firm. Anyone who's smart enough can develop these skills because the open source software is available to people outside the firm.

Hiring and firing becomes easier because there's a larger labor pool to draw from, and switching costs between employees are lower compared with the closed source situation. Given the natural imbalance between employers and employees, this aspect of open source is likely to increase competition for jobs and drive down salaries. Lower salaries aren't as much of an advantage to the software vendor as might be expected because in the more transparent and competitive open source situation, such cost savings are likely to be (at least partially) passed on to customers.

3.4 An Overview of Open Source Business & Community

Business models can create value either by enhancing the customers' willingness to pay or by decreasing suppliers' and partners' opportunity costs—for example, through improved transaction efficiency. The total value created by a business model is also a function of the competitive alternatives, in other words, the market power of the focal firm's business model vis-à-vis rival business models. The total value created is the value created for all business model stakeholders (focal firm, customers, suppliers, and other exchange partners). It is the upper limit for the value that can be captured by the focal firm [38].

How does business model design influence the competing claims to total value created by different stakeholders in the business model? Drawing on Brandenburger and Nalebuff [39] and Brandenburger and Stuart [38], we reason out that there is a positive association between the design of the business model and the performance of the focal firm—if, for a given level of competition, the focal firm's business model design creates value, and it does not decrease its bargaining power relative to other business model stakeholders.

Open source software products provide access to the source code (or basic instructions) in addition to executable programs, and allow for this source code to be modified and redistributed. This freedom is a rarity in an industry where software makers zealously guard the source code as intellectual property.

In making the source code freely available, a large number of developers are able to work on the product. The result is a community of developers spread around the world working to better a product. This approach has led to the popular operating system Linux, which has emerged as a credible threat to Microsoft's products—especially on the server side. Other famous open-source products include Apache (a program used to run Web sites), OpenOffice (an alternative to Microsoft Office), and Sendmail (the program that facilitates the delivery of approximately 80 percent of the world's e-mail).

Open source is typically viewed as a cooperative approach to product development, and hence more of a technology model. It is typically not viewed as a business approach. However, increasingly we find that entire companies are being formed around the open source concept. In a short period of time, these companies have amassed considerable revenues (although it is fair to say that most of these firms are not yet profitable).

All software companies exist to make maximum profits. Therefore, it is common for these corporations to seek out new ways of generating revenues and reducing costs. Increasingly, companies are using open source as a business strategy to achieve both these objectives.

On the cost reduction side, software producers are now able to incorporate the source code from an open source product into an existing code base. This allows them to reduce the cost of production by reusing existing code. For example, Microsoft, the world's largest software maker, has used source code from a leading open source operating system (Berkeley System Distribution or BSD) in its Windows 2000 and XP products and has acknowledged this on a public web site.¹ It is becoming more common for companies to forge strategic alliances with communities of open source software developers. The community develops the product and thus

reduces the cost burden on the company. A prime example of this is the strategic alliance between Ximian and Microsoft in building a connection between the Net initiative and Linux.²

On the revenue side, some open source products are now in such great demand that there is a strong need for support services for enterprise customers. These support services includes installation, training/certification, and ongoing technical assistance. Service contracts for these products have become a strong revenue source for companies such as Red Hat Linux.

From the consumer perspective, open source products are attractive due to their reduced cost and comparable performance. Governments, for example, are increasingly motivated to adopt open source products to reduce the expenditure of scarce taxpayer money. Some governments (such as Argentina and Peru) have experimented with moving entirely to an open-source model.

Even for individual consumers, open source products are becoming accessible. Wal-Mart has started to carry PCs that run Linux. Many free applications are now available for PCs. For example, OpenOffice and KOffice are free, open source products that directly compete with Microsoft's Office suite.

3.4.1 Producers of Open Source Products—the Community

The producers of open source products (figure 3.4) are typically a diverse group of developers with a shared passion for a product. They do not seek a profit and do not distinguish between corporate and individual users.

Therefore, they make (a) the product and (b) the source code available for free to any interested user. There is usually support available through electronic mailing lists and Usenet groups. Members participate to learn more about the product and believe that others will help them if they have a need [40]. Surprisingly, the customer support provided by communities surrounding products such as Apache and Linux have won awards for excellence.

1. See <http://support.microsoft.com>. Knowledge Base article 306819.

2. Ximian is now owned by Novell.

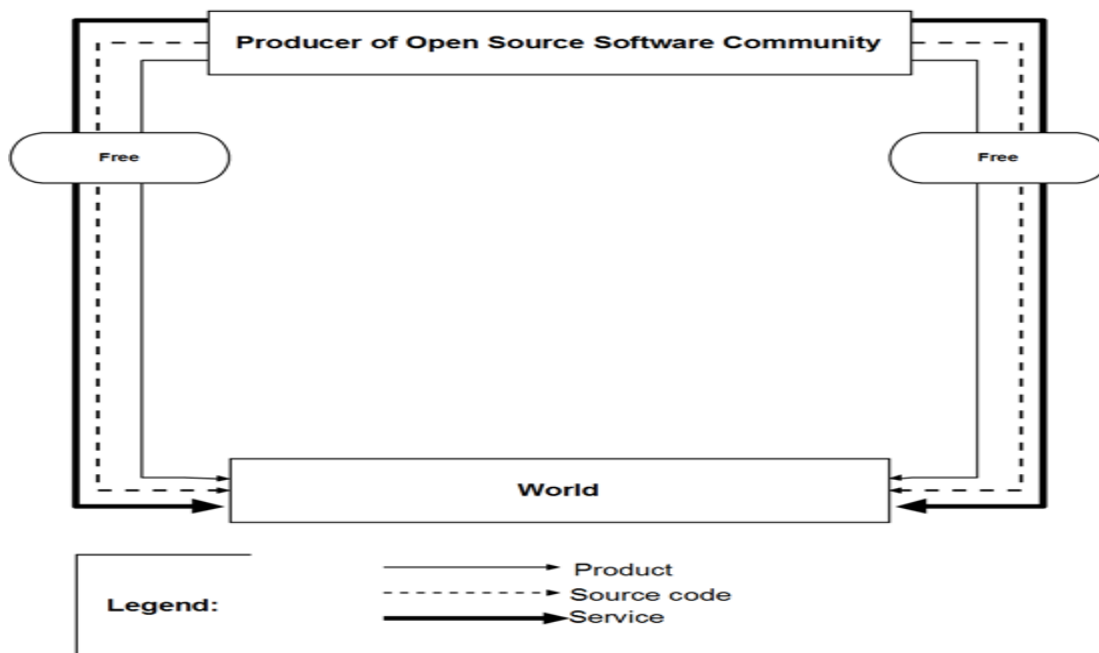


Figure 3.4 the producers of open source products

The community of producers is frequently portrayed as being inimical to corporate profits. However, I submit that the community is simply *indifferent* to its own profits as well as profits that any corporation can make from its products. Open source developer communities are frequently interested in adoption of the product by the intended target audience. Importantly, they want any interested developer to have access to the entire code so that the person can tinker with it to make improvements.

There is no sense of direct competition with companies. A company that views a community as its competitor is welcome to look at its entire source code, whereas the opposite is never true. Communities do not distinguish between users across countries. When the product is available for free, it is amazingly easy to make a product global. There is no issue of taxation or piracy.

The community controls what happens with the product by making one crucial choice—the license. The original developers control the copyright for the intellectual property at all times. However, there is considerable variation between licenses with regard to how derived works may be distributed.

There are a number of licenses from which communities can choose. However, they can be broadly classified as the GNU General Public License (GPL) versus everything else. The GPL is the most famous license and products such as Linux are distributed using it. The key feature of the GPL is that it restricts the terms of distribution of derived works. If a company incorporates GPLed source code in its products, it must make the source code for any product it sells in the marketplace available to any interested party under the terms of the GPL. This provision frightens corporations interested in selling open source products. However, it is important to note that there is a whole host of other licenses that do not have this stipulation.

In my view, the derived works clause is so powerful that it affects how business models are constructed. The discussion about business models is therefore broken down into the GPL and the non-GPL model. Generally speaking, use of GPL reduces the profit potential of companies.

It is very important to note that the open source community does not set a price on a software product. Even in the case when the product is available for free, anybody can incorporate the product and sell it for a price. Even with a GPL license this is possible. Obviously, in the case of GPL, there is the attendant duty of making the source code for derived works freely available.

3.5 Business Models

The business model is the way products/services are sold to customers, cash is generated, and income is produced. Not only do OS producers assemble viable programs using code available on the Internet, maintain relations with the community, and offer reliable and transparent pieces of software, but they also challenge the traditional business model based on selling proprietary software through licenses.

In this section, we discuss the various business models built around the open source philosophy. It is certainly true that some companies benefit from the sale of hardware that runs open source products [41]. Similarly, the market for embedded products can be great. However, for the purposes of this chapter, we focus on the software and service-oriented business.

3.5.1 The Distributor

The distributor provides access to the source code and the software. In the case of Linux, leading distributors include Red Hat, Suse, Ubuntu, and Linux mint. Distributors make money in these ways:

1. Providing the product on CD rather than as an online download—most people are not comfortable with downloading the product from a Web site. Therefore, there is money to be made selling the product in CD form. According to one source(<http://www.distrowatch.com>), as of November 2011, the highest price that was being charged for a Linux CD was \$21.95 (Linux Mint) and the lowest price for a CD was zero (for instance, Debian and Gentoo).
2. Providing support services to enterprise customers—enterprises are willing to pay for accountability. When they have a problem, they do not want to send a message to a mailing list and wait for support that may or may not be of the highest quality. They have no interest in sifting through technical FAQs to find the answer. Therefore, there is money to be made in services such as support for installation, answering technical questions and training employees to use the product.
3. Upgrade services—in which enterprises can now enter into long-term agreements with distributors to ensure that they get the latest upgrade. By acting as application service providers, distributors can help their clients get the latest version of the product seamlessly .The business model of distributors is shown in figure 3.5.

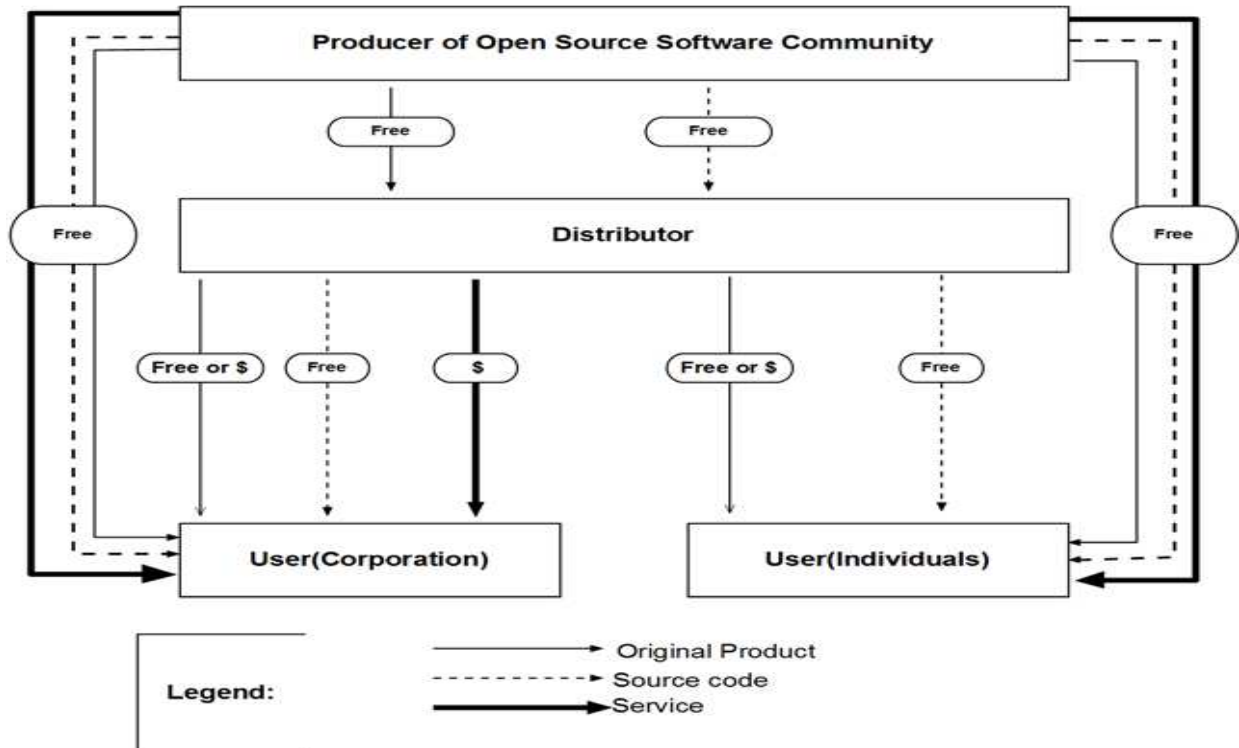


Figure 3.5: The Distributor Business Model

3.5.2 The Software Producer (Non-GPL Model)

Software producers can benefit from the open source software community in two ways. First, they can incorporate the source code of an existing product in a larger code base and create a new product. Second, they can also take an entire open source product and bundle it with existing products. (I am using the term *derived product* in a very general sense here to include both these cases.) The source code for the derived product does not need to be disclosed, because the license is not GPL.

As mentioned earlier, Microsoft has incorporated the code from BSD in its products and has not released the source code to any interested party. All Microsoft had to do was to acknowledge that it benefited from BSD's code.

Using this business model the software producer benefits from lowered cost of production and hence increased margin, in this case. There is a service revenue stream in place here as well. The business model itself is shown in figure 3.6.

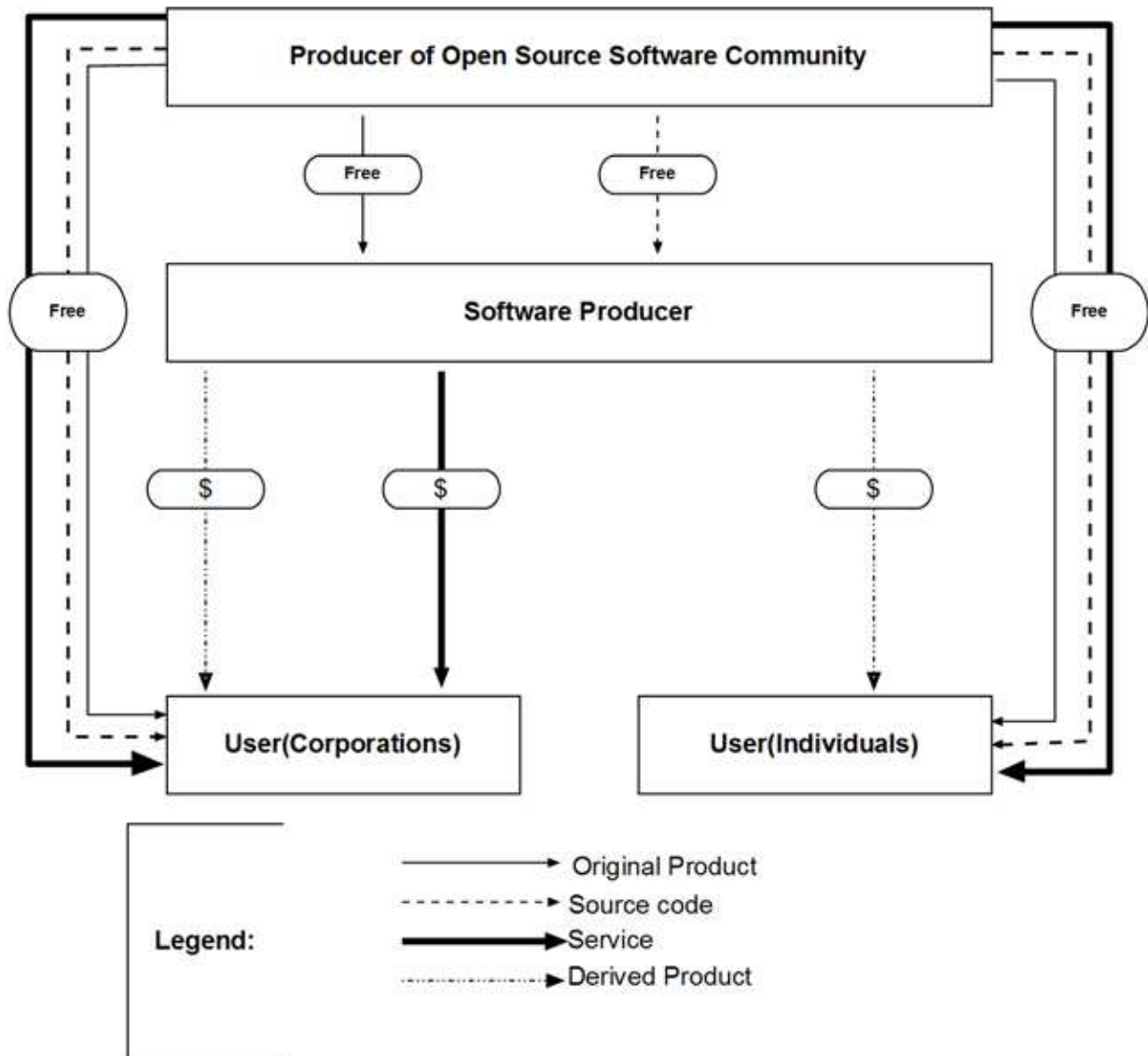


Figure 3.6: The Software Producer (Non-GPL Model) Business Model

Interestingly, the source code for the original product is still available to the end users from the community. In the cases where the derived product is a small adaptation of the original product, this may be very useful to the end users. This is the cost the for-profit software producer pays to get the source code for free.

3.5.3 The Software Producer (GPL Model)

The key difference between figures 3.6 and 3.7 is that in the latter, which shows the business model for this case, the software producer is forced to make the source code for the derived product available to the end user.

Let us compare the GPL and non-GPL models. The release of the source code in the GPL model accelerates innovation, due to more rapid feedback and input. Greater inclusion of users builds relationships, and hence loyalty. Also, if the user builds a new version of the product for commercial use, the company gets to see it along with the source code. However, it does expose the inner workings of the company's product to the users.

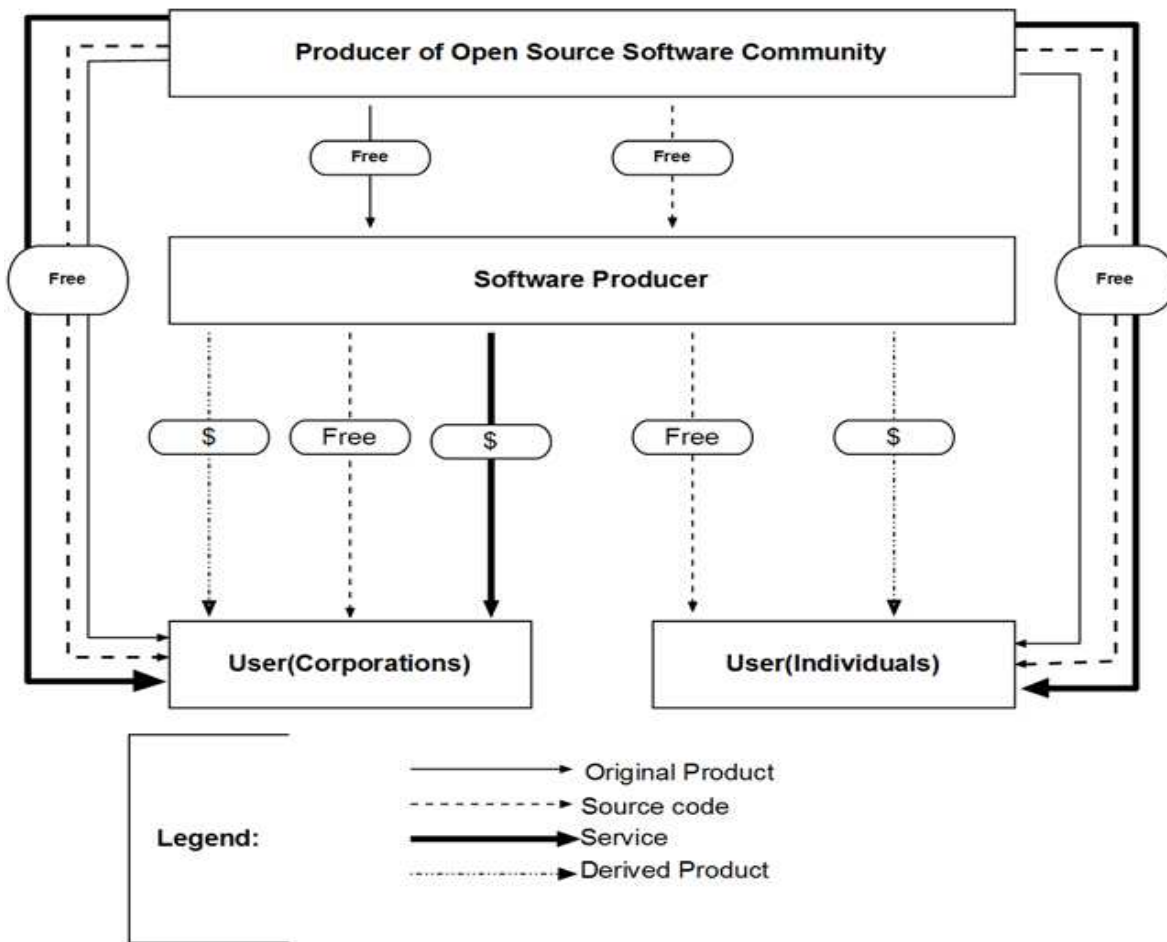


Figure 3.7: The Software Producer (GPL Model) Business Model

Ultimately, the difference between the GPL and non-GPL models is in terms of what the seller expects from the user. The GPL software producer expects an empowered user who is eager to engage in a two-way conversation. The non-GPL software producer wants the recipient of the software to simply use it and do nothing else.

3.5.4 The Third-Party Service Provider

The mission of third-party service providers is simple. They don't care where you got the code or where you got the product. If the product you are using meets a broad set of criteria, they will fully support it. They have one single revenue stream—service. Their business model is shown in figure 3.8.

Why should users—especially corporations—use these providers? The bottom line is that paid service generally equates to higher-quality service. Moreover, in many cases, third-party service providers are local and may therefore be able to provide onsite assistance that is typically impossible in the case of free service on mailing lists and user groups. It is important to keep in mind that these service providers are competing with the community to provide customer service.

I have presented two types of models here—one in which the company sells software and service and one in which a company simply offers a service. It is interesting to speculate on whether a company can survive on the sale of software alone.

How can a company add value? First, it can choose a version of the product that is stable and that is most suited to its users' needs. Second, it can create a suite of products that are well integrated. These products may come from different sources—some open source, some commercial. The value addition is in creating one package that works well together.

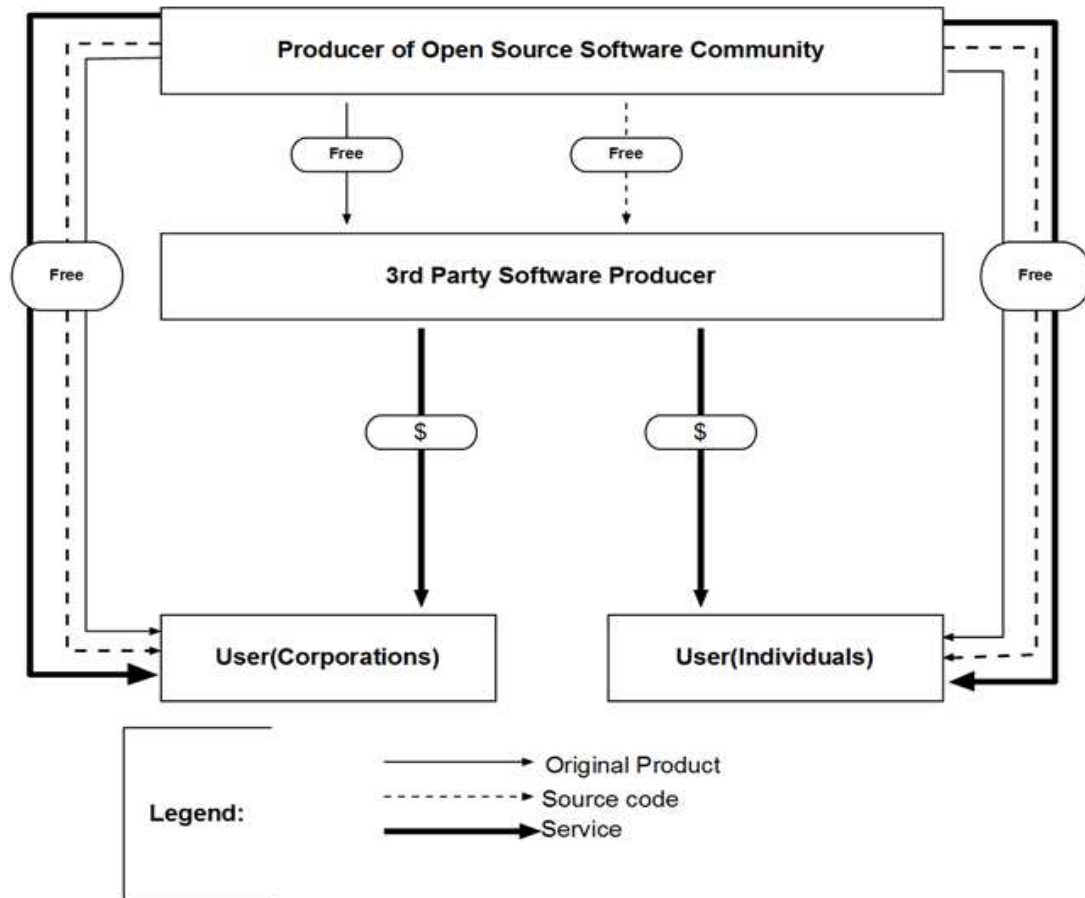


Figure 3.8: Third Party Service provider Business Model

In general, we find that sale of software alone is insufficient to sustain a business. What is needed is software and service. For many software sellers, they already have a relationship with enterprise customers. They can benefit most by up-selling—that is, selling more to existing corporate customers. Selling service then becomes a logical conclusion. Even with commercial software, all software sellers use service as a strong secondary revenue stream.

Beside those above business models we found that few firms choose a pure business model by offering OSS solutions alone [15]. We called this Mainly OSS business model. Again, The vast majority combine proprietary and OSS products and receive revenues from traditional license fees as well as from OS-related services are hybrid business model [15]. Firms' that using code from OSS community and also using their own code too but they don't supply code for their derived solution and making revenue are following Proprietary business model. Firms' those

following only proprietary model to make revenue we classified them as Mainly Proprietary solution provider.

We also see some other business models for companies who want to take advantage of open source software, and evaluate their potential success. Zoe konovalov [42] provide some business model those focus mainly on reducing costs of software as an input and that create profits from associated benefits of open source software.

1. The Apache model: cost-sharing. This business model relies on open source development not as a source of profits, but as a way to drastically reduce the costs for companies, allowing their primary businesses to become more profitable. If different users all have a similar need for a complex type of software that is a primary input to their business, they can derive great benefits from the cost of development and reaping the advantages of widespread peer review.

2. Risk-spreading. This is another form of cost reduction, in which the encouragement of an open source community reduces the possible risk of a particular developer leaving or going out of business.

3. Market Positioning. This is the use of open source development in one software market to entrench a position and allow for the sale of proprietary software in an associated market. One example is Netscape releasing the source code of its Mozilla browser in an attempt to keep Microsoft from gaining a browser monopoly, so that it could continue to hold a viable place in the server market.

4. Encourage the sale of complementary hardware. This was the earliest open source business model, since IBM originally released all the source code for the operating systems of its early mainframes. Hardware and software are complementary goods; a fast chip is useless without software make it work. Unlike software, hardware is a rivalry good: if you own a computer that precludes you from the same computer. It is therefore very simple to capture sale value from hardware, and open source software can be used as a way of increasing the value of that hardware. Hardware companies such as IBM, Hewlett-Packard, and Sun Microsystems have all

announced various kinds of open source initiatives and alliances with the Linux community, so it is clear that this business model is an important and viable one.

5. Sell services associated with open source software. The old cliché would be “Give away the razor, sell the razor blades,” although this is not completely precise since a razor is a rivalry good while open source software is not. A better analogy might be “Give away the recipe, open a restaurant.” Companies following this business model include RedHat, which sell well-designed packages of ready-to-install versions of Linux, offers installation support and technical services.

6. Intellectual capital marketing. Prestigious consulting companies such as McKinsey and the Boston Consulting Group publish journals collecting their most important market research and business insight. It is true that these allows access to their work without paying a consulting fee, but there are more than repaid by the marketing values of these journals, increasing the companies’ intellectual reputation and their name recognition. In a similar way, a popular open source release, if associated with a particular company, acts like free advertising for the expertise of its human capital- its programmers and can allow it to gain contracts for individual customization.

Several business models have been proposed for open source ([43]; [44]; [45]; [46]), but little attention has been paid to dual licensing. Jesper Holck, Roberto Zicari, and Volker Mahnke [47] have focus of these issue. They have identified two classes of open source software business models which have the potential for success: Commodity Products for Horizontal Markets (CPHM) and Innovative Products for Niche Markets (IPNM).

The two business models are, however, differentiated as follows: The *CPHM* strategy is based on commodity products in horizontal markets, with large communities; the strategy is to consolidate or even expand the horizontal market. The *IPNM* strategy is based on innovative products in niche markets, with smaller communities; the strategy is to aggressively expand from niche to horizontal market, when market conditions make this possible. Interestingly, the same firm may use both business models for different products. E.g., the regular MySQL product can be seen as a highly popular commodity, but MySQL Cluster and other special versions are technology and innovation leaders [48].

The first results of applying this approach to a real case study can be found in [49], where we discovered that MySQL, using a combination of series of rapid incremental innovation, stack-based complement strategies, an open-source product testing approach, and a dual licensing strategy succeeded in the increasingly commoditized database software markets.

Summary

We now know that it is possible to build a business around the open source strategy. We are increasingly finding that open source software communities are awesome competitors. They are able to compete with large companies on an equal footing and even defeat them. They are, therefore, not to taken lightly or dismissed offhand.

Open source software is not for hobbyists any more. Instead, it is a business strategy with broad applicability. Businesses can be built around this idea. And business model have significant impact on earning revenues.

To this end, I have proposed three fundamental business models: distributor, software producer (GPL and non-GPL), and the third-party service provider. These are sustainable models that can lead to robust revenue streams. The business models provided here can be enhanced by the addition of further revenue streams. For instance, we now know that certification of developers on an open source product can lead to strong revenues.

The future of commercial open source software lies in commercial business strategies, but which are the strategies that are more likely to deliver the results vendors are looking for?

4 Data Gathering

4.1 Data

This empirically oriented research uses a unique data of the software industries from a multi-country study ELISS survey.

This database was developed in 2004 by the Laboratory of Economics and Management at the Sant'Anna School of Advanced Studies within the CIPR project of the PRIME Network of Excellence, which was funded by the European Community within the Sixth Framework Program. ELISS provides information on OSS firms. Data were collected through a structured questionnaire³ administered to the owner-managers and chief technology officers of a random sample of 6,000 firms operating in the software industry (NACE code 72.2) and located in five European countries (Finland, Germany, Italy, Portugal, and Spain). The sample is stratified by firm size and geographical area (NUTS2 level). It defines as OSS firms' those firms that supply, OSS-based products and services to their customers. It is worth noting that a firm is defined as OSS even if its offering includes also proprietary solutions.

The companies that answered the questionnaires were 915 and the response rate was 15.3%. The questionnaires included data on the business model, the internal organization of the companies (share of graduate employees, ownership structure etc.).

The sample is composed of all the companies that have accounting performance data for at least two financial years. Accounting data on companies located in Finland, Italy, Portugal and Spain for the period 2004-2009 were recently added, using the AMADEUS and ORBIS databases accounting data like the financial data- sales, number of employee, tangible assets, intangible assets, total assets, and salaries of those firms were extracted. I would also like to explain that we made a search by VAT code. Even though, German firms were in the ELISS database but we don't took those firms in our empirical analysis consideration due to the lack of VAT code.

Our study focused on a hidden population. Indeed, no official census exists of the OSS firms located in the five European countries we considered. The absence of a precise definition of the population of OSS firms did not allow us to extract a representative sample. Notwithstanding this limitation, our sample exhibits several important strengths. First, it includes a large number of OSS firms located in different countries, while samples used in prior quantitative studies on OSS firms were smaller and only included OSS firms from one country [50]. Second, our sample exhibits substantial heterogeneity with respect to the variables of sales, number of employee, tangible assets, intangible assets, total assets, and salaries, the number of OSS-based activities in which the OSS firms are involved, firm size and the type of tasks (i.e., low-value-added tasks vs. high-value-added tasks) that OSS firms carry out.

Country	Frequency	Percent
Finland	117	30.95
Italy	99	26.20
Portugal	46	12.16
Spain	116	30.69
Total	378	100%

Table 1 Country Distribution of Firms

Table 1 reports some descriptive statistics on the 378 sample firms. The sample firms are not equally distributed across countries. The sample dataset comprises of firms of which about 31% firms are located in Finland, 26% are Italian, 12% are Portuguese and more than 30.5% are located in Spain.

3 Questionnaires are available in the appendix.

	No. of firms	%
<i>Year of foundation</i>		
<1985	39	10.54
1985-1989	67	18.19
1990-1994	65	17.58
1995-1999	105	28.38
≥2000	94	25.31
Total	370	100 %
<i>Sales at 31st December 2003</i>		
≤2,000 k€	226	74.85
2,001 – 10,000 k€	58	19.20
10,001 – 50,000 k€	17	5.62
> 50,000 k€	1	0.33
Total	302	100%
<i>No. employees at 31st December 2003</i>		
0-10	147	56.54
11-50	75	28.85
51-250	27	10.38
> 250	11	4.23
Total	260	100%
<i>Total Assets at 31st December 2003</i>		
≤1,000 k€	220	91.67
1,001 – 5,000 k€	16	6.67
5,001 – 10,000 k€	2	0.83
> 10,000 k€	2	0.83
Total	240	100%

Table 2 Distribution of the sample firms By Year of Foundation and Size

Table 2 reports that, thirty-nine sample companies were established before 1985. About, 18% of the sample companies were established between 1985 and 1989. In the same way 65 companies were founded in the period of 1990 to 1994. Most of the sample companies about 54% were founded 1995 to onwards.

We analyzed size distribution by distinguishing micro firms', small firms', medium firms and large firms'. Most sample firms are micro or small. In the case of sales, in the financial year 2003 Two hundreds twenty-six (226) companies had a sale of around 2000K€, which about 75%. Again, 19.20% companies had sales in the range of 2001-10000k€, which in the number, 58

companies. Seventeen companies had sales of 10001-50000k€. Only one of the firms had sales more than 50,000 k€ .It seems that most of the companies doing OSS business are small and medium scale.

In the sample data set, we see most of the firm, which is about 56%, had employed less than 10 people. In the number, it is 147 sample companies. 75 sample firms employed 10 to 50 people. In the percentage of companies, it is about 29% of the sample. Twenty seven companies employed 51 to 250 people. Only eleven companies had more than 250 employees. Quite remarkably, most sample firms are small: 56% of them have less than 10 employees, freelancers, and owner-managers.

Table 2 also reports that, most of the sample companies have total assets less than 1 million euro at the end of year 2003.In percentage it is about 91.67% and only 16 samples companies had total assets in the range of 1million to 5 million euro's. From these accompany data it apparent that companies doing OSS business are small and medium scale.

5 Methodology of the econometric analysis

The aim of the econometric analysis was to find out the impact of the business model on firms' growth performance. We have discussed about business model in chapter 3 and saw firms' following various business strategies to make revenue. At the beginning of this thesis, our aim was to make an empirical analysis of those entire business models over their performance. But due to the lack of required data to perform this analysis, we set out our new goal to analyse the performance of the firms' doing business with 1) Mainly OSS solution 2) Hybrid Firms 3) Mainly proprietary solution, this typology were discussed in [15] and firms' are more OSS (MOSS) oriented vs. less OSS (LOSS) oriented [15]. We have distinguished those typology in chapter 3.

5.1 The specification of the econometric model

In the econometric analysis, we use FP as a measure of firms' economic performance. To investigate whether choosing a business strategy (BS) has an impact on Firms' performance (FP), the following econometric model (Gibrat law) is specified:

$$\ln FP_{i,t} = \alpha + \beta' D \text{var}_{i,t} + \gamma' BS_{i,t} + \lambda Age_{i,t} + \delta' Z_{i,t} + \eta_i + \varepsilon_{i,t}, \quad (1)$$

where $\ln FP_{i,t}$ is a measure of firms' performance (described in detail in Section 5.2); $BS_{i,t}$ is a vector of covariates that captures the number and characteristics of the BSs in which sample firms were involved at time t ; $D \text{var}_{i,t}$ is a vector of dependent variables measuring the performance of firms'; $Age_{i,t}$ denotes the age of firms at t '; $Z_{i,t}$ are a series of control variables that include firm-specific attributes; η_i are the unobserved firm-specific fixed effects; and $\varepsilon_{i,t}$ are the usual independent and identically distributed (i.i.d.) disturbance terms.

In this paper, we aim to assess the impact of the Business Strategies (BS) on the firms' performance (FP). In so doing, we face two obstacles. First, a positive association between firms' performance (FP) and the formation of specific types of Business Strategies (BS) may simply be a consequence of reverse causality, with higher FP firms' being more likely to establish these alliances. Second, there may be unobserved factors such as smart management that explain both firms' economic performance and their involvement in a particular Business Strategies (BS). To account for possible time-varying and unvarying unobserved heterogeneity that, if correlated with regressors, may lead to biased estimates of the parameters of interest, GMM is used for estimation.

More specifically, to take into account problems generated by the potential endogeneity of the explanatory variables, we resort to the GMM-system (GMM-SYS) estimator developed by Blundell and Bond [52]. Accordingly, we simultaneously estimate first difference and level versions of Eq. (1). As the instruments, we use appropriately lagged values of the endogenous variables for first difference estimates and appropriately lagged differences for level estimates. This method alleviates the typical problem of weak instruments characterizing the first GMM-difference (GMM-DIF) estimator.⁴ Moreover, it preserves information from the level equations enabling consistent identification of time-invariant covariates. As is customary in this type of analysis (e.g. [53]), we formulate the weakest possible assumption.

5.2. Dependent variable

We use SALES, HEADCOUNT and TOTAL ASSETS to measure firms' growth performance. These are typical indicators of firms' growth. Whereas SALES is the sales of a firm in a financial year, HEADCOUNT is the number of employees of a firm on a particular year and TOTAL ASSETS is the assets of a firm in a particular financial year. Those are calculated as logarithm of respectively, SALES+1 in k€, TOTAL ASSETS in k€ and HEADCOUNT+1. This indicator of firm growth performance is particularly appropriate in the context of our analysis.

⁴ In our case, pseudo-first stage regressions support our choice of the GMM-SYS estimator, highlighting that lagged instruments in first differences are strongly correlated with Business strategies variables; on

the contrary, lagged instruments in levels are poorly correlated with the change in Business mode covariates pointing to the strength other than the validity of the additional instruments implied by the GMM-SYS estimator with respect to GMM-DIF.

5.3. Explanatory variables

Definitions of all the explanatory variables used in the estimation of the econometric model are reported in Table 3.

Variable	Definition
Age _t	Age of the firm at year t
Headcount _{t-1}	Number of employees <i>i</i> participate at year <i>t-1</i>
Sales _{t-1}	Log of sales+1 of the firms at year <i>t-1</i>
Total Assets _{t-1}	Log of total assets+1 of the firms at year <i>t-1</i>
DMainlyOSS	Participant providing Mainly OSS solution at year <i>t</i>
DMainlyProp	Participant providing Mainly Proprietary solution at year <i>t</i>
Software_solution	Types of Solution provided by the firms.
Age1	Age of a participant at year t from the foundation year.
MOSS	Dummy variable =1 for more OSS oriented firms

Table 3 Explanatory variables used in the estimation of the econometric model.

The dependent variable (DMainly OSS) is a dummy equal to 1 for firms' that mainly offer OSS based solutions to customers and 0 otherwise. A different variable (DMainly Prop) is a dummy that is equal to 1 for firms' that mainly offer proprietary and 0 otherwise. One more dependent variable (MOSS) is a dummy that is equal to 1 for firms' that adopted this organizational practice and 0 otherwise. This question was posed only firms' they are mostly OS oriented.

Firstly, we tested the performance of DMainly_OSS firms' over DMailnlyProp by including among the regressors a measure of the size of firm's operations (lnheadcount) calculated as the

logarithm of the total number of the firm’s owner-managers, employees, and freelancers in t year and (lnheadcount1) in t-1 year, sales of the firms(lnsales) calculated as the logarithm of the firm’s sales in t year and (lnsales1)in t-1 year and total assets (Intotalassets) calculated as the logarithm of the firm’s sales in t year and (lnassets1) in t-1 year. The results are shown in table 6, 7 and 8 respectively.

Secondly, we tested performance of MOSS firms’ by including among the regressors a measure of the size of firm’s operations (lnheadcount)calculated as the logarithm of the total number of the firm’s owner-managers, employees, and freelancers in t year and (lnheadcount1) in t-1 year, sales of the firms(lnsales) calculated as the logarithm of the firm’s sales in t year and (lnsales1)in t-1 year and total assets (Intotalassets) calculated as the logarithm of the firm’s sales in t year and (lnassets1) in t-1 year. We predicted a positive sign for lnheadcount, lnsales and Intotalassets. The results are shown in table 9, 10 and 11 correspondingly.

Variable	Number of observations	Mean	S.D.	Min.	Max.
Headcount	2008	176.763	2277.002	0	38498
Sales	2311	6453850	5070000	0	915000000
Total Assets	2020	580130.4	3239894	0	58400000
DMainlyOSS	2591	.132	.339	0	1
DMainlyProp	2595	.698	0.459	0	1
Software_solution	870	2.469	1.005	1	5
Age1	2628	10.523	7.121	0	38
MOSS	820	.583	.4933	0	1

Table 4 Descriptive statistics of explanatory variables of econometric models.

	Age1	headcount	sales	totalassets	DMainlprop	DMainLOSS	software_sol	Moss
Age1	1.000							
headcount	0.005	1.000						
sales	0.100	0.803	1.000					
totalassets	0.120	0.552	0.805	1.000				
DMainlyProp	0.107	-0.130	-0.091	-0.095	1.000			
DMainlyOss	-0.266	0.201	0.131	0.116	-0.591	1.000		
Software_sol	-0.284	0.089	0.025	0.064	-0.591	0.811	1.000	
Moss	-0.102	0.086	0.043	0.025	-0.030	0.575	0.557	1.000

Table 5 Correlation matrix of the explanatory variables included in the model.

Table 4 gives descriptive statistics and Table 5 gives the correlation matrix of the explanatory variables included in the models. Correlations among variables were low, so we excluded the presence of multi-collinearity problems.

5.4. Econometric results

The results of the econometric analysis are illustrated in Tables 6 to 11. We predicted a positive sign for lnheadcount, lnsales and lntotalassets. The results are shown in table 6, 7 and 8 respectively.

Inheadcount	Coef.	Corrected Std. Err.
Inheadcount1	.633	(.216) **
DMainlyOSS	.228	(.134) *
DMainlyProp	.551	(.276) **
age1	.004	(.005)
GraduateShare	.508	(.249)**
OwnerStructure	.168	(.139)
_cons	-.191	(.149)
Wald chi2(6) = 265.10		Number of obs = 299

Table 6 DMainlyOSS performance over Dmainlyprop in terms of lnheadcount

** , * , indicate Significant at the 5%, and10% level of significance.

Insales	Coef.	Corrected Std. Err.
Lnsales1	.458	(.058) ***
DMainlyOSS	.971	(.496) **
DMainlyProp	.874	(.264) ***
age1	.006	(.015)
GraduateShare	.540	(.193) ***
OwnerStructure	.202	(.125)
_cons	6.185	(.714)
Wald chi2(6) = 114.70		Number of obs = 366

Table 7 DMainlyOSS performance over Dmainlyprop in terms of Insales

***, ** indicate Significant at the 1% and 5% level of significance.

Intotalasstes	Coef.	Corrected Std. Err.
Intotalasstes1	.930	(.117) ***
DMainlyOSS	.559	(.240) **
DMainlyProp	.475	(.133) ***
age1	-.002	(.013)
GraduateShare	.195	(.268)
OwnerStructure	.005	(.095)
_cons	.323	(1.018)
Wald chi2(6) = 524.84		Number of obs = 329

Table 8 DMainlyOSS performance over Dmainlyprop in terms of Intotalassets

***, ** indicate Significant at the 1% and 5% level of significance.

Inheadcount	Coef.	Corrected Std. Err.
Inheadcount1	.907	(.081) ***
moss	.209	(.066) ***
age1	-.001	(.003)
GraduateShare	.081	(.132)
OwnerStructure	.165	(.099) *
_cons	-.065	(.068)
Wald chi2(5) = 1726.36		Number of obs = 370

Table 9 MOSS performance in terms of Inheadcount.

***, * indicate significant at the 1% and 10% level of significance.

Insales	Coef.	Corrected Std. Err.
Insales1	.258	(.067) ***
moss	2.054	(.473) ***
age1	.038	(.017) **
GraduateShare	-.025	(.510)
OwnerStructure	1.361	(.513) ***
_cons	7.647	(.801)
Wald chi2(5) = 98.19		Number of obs = 407

Table 10 MOSS performance in terms of Insales.

***, ** indicate significant at the 1% and 5% level of significance.

Intotalassets	Coef.	Corrected Std. Err.
Intotalassets1	.728	(.088) ***
moss	1.055	(.364) ***
age1	-.001	(.013)
GraduateShare	.094	(.278)
OwnerStructure	.324	(.211)
_cons	2.137	(1.074)
Wald chi2(5) = 98.79		Number of obs = 389

Table 11 MOSS performance in terms of Intotalassets.

*** indicate significant at the 1% level of significance.

Our findings showed that, as to headcount grow: 1) Both DMainlyOSS and DMainlyProp are positive and significant in all the models. This indicates that hybrid firms' are those that grow less. 2) MainlyProp grow more than OSS firms' (be the mainly OSS or Hybrid) as the positive and significant coefficient of DMainlyProp greater than the coefficient of DMainlyOSS, shows OSS firms' grow less than proprietary ones in terms of employee as the former may use the work of OSS community developers hence they do need less employees. Again, in terms of both sales and total assets MainlyOSS grow more than the remaining firms'. If we distinguished MOSS vs. LOSS we also see a positive outcome.

6 Conclusion

The rise of the OS movement attracted the interest of economists and organizational scientists from the outset, mainly because it is a challenge to received theories of individual motivation and task coordination.

This empirical analysis has investigated the relationship between business models and performance of the software companies in five European countries. Empirical analysis reported in this article has provided some new pieces of information on the mechanisms of the open source software supply and licensing and the software firms' role in that. Further systematic theoretical and empirical economic analyses on firms' choices and behavior in situations when the open source products are supplied along side with their proprietary substitutes are definitely needed.

Stated simply, strategy or business model is a road map or guide by which an organization moves from a current state of affairs to a future desired state. It is not only a template by which daily decisions are made, but also a tool with which long-range future plans and courses of action are constructed. Strategy allows a company to position itself effectively within its environment to reach its maximum potential, while constantly monitoring that environment for changes that can affect it so as to make changes in its strategic plan accordingly. In short, strategy defines where you are, where you are going, and how you are going to get there.

All business decisions are fundamentally based on some set of values, whether they are personal or organizational values. The implication here is that since the strategic plan is to be used as a guide for daily decision making, the plan itself should be aligned with those personal and organizational values. To delve even further, a values assessment should include an in-depth analysis of several elements: personal values, organizational values, operating philosophy, organization culture, and stakeholders. This allows the planning team to take a macro look at the organization and how it functions as a whole. Implementation of the strategic plan is the final step for putting it to work for an organization. To be successful, the strategic plan must have the support of every member of the firm.

We acknowledge that this paper has some limitations that open opportunity for future research. I would say that different classifications of business models may be taken into consideration in future works. However, this result contributes to:

- 1) The literature on OSS as they investigate a topic that has never been explored before.
- 2) The literature on business model as they provide further insight on the performance impact on business model choice. In terms of both sales and total assets, we see from the empirical analysis MainlyOSS grow more than the remaining firms'.

Bibliography :

- [1] Wafa Hakim Orman, Giving It Away for Free? The Nature of Job-Market Signaling by Open-Source Software Developers-2008
- [2] Heli Koski, Private-collective software business models: coordination and commercialization via licensing.
- [3] R. Goldman and R. Gabriel, Innovation Happens Elsewhere, Morgan Kaufmann, 2005.
- [4] Steven Weber “The Political Economy of Open Source Software”, 2000.
- [5] Levy, Stephen, Hackers: Heroes of the Computer Revolution. New York: Dell, 1985.
- [6] Leonard, Andrew, the Free Software Story. <http://www1.salon.com/tech/special/opensource/>
- [7] <http://www.apache.org/>
- [8] E. S. Raymond, The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, O’Reilly, Sebastapol, CA, 2001.
- [9] E. S. Raymond, Up from Alchemy, IEEE Software 21 (1) (2004) 88, 90.
- [10] A. Mockus, R. T. Fielding, J. D. Herbsleb, Two case studies of open source software development: Apache and Mozilla, ACM Transactions on Software Engineering and Methodology 11 (3) (2002) 309–346.
- [11] K. Crowston, J. Howison, The social structure of free and open source software development, First Monday 10 (2).
URL <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1207/1127>
- [12] W. Scacchi, Free and Open Source Development Practices in the Game Community, IEEE Software 21 (1) (2004) 59–66.
- [13] E. von Hippel, G. von Krogh, Open source software and the “private-collective” innovation model: Issues for organization science, Organization Science 14 (2) (2003) 209.
- [14] G. Hertel, S. Niedner, S. Herrmann, Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel, Research Policy 32 (7) (2003) 1159 – 1177, open Source Software Development.
- [15] A. Bonaccorsi, S. Giannangeli, C. Rossi, Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry, Management Science 52 (7) (2006) 1085–1098.

- [16] C. Gacek, B. Arief, The Many Meanings of Open Source, *IEEE Software* 21 (1) (2004) 34–40.
- [17] K. Crowston, Q. Li, K. Wei, U. Y. Eseryel, J. Howison, Self-organization of teams for free/libre open source software development, *Information and Software Technology* 49 (6) (2007) 564 – 575, qualitative Software Engineering Research.
- [18] L. Yu, Self-organization process in open-source software: An empirical study, *Information and Software Technology* 50 (5) (2008) 361 – 374.
- [19] K. R. Lakhani, E. von Hippel, How open source software works: 'free' user-to-user assistance, *Research Policy* 32 (6) (2003) 923 – 943.
- [20] S. K. Sowe, I. Stamelos, L. Angelis, Understanding knowledge sharing activities in free/open source software projects: An empirical study, *Journal of Systems and Software* 81 (3) (2008) 431–446.
- [21] L. Zhao, S. Elbaum, Quality assurance under the open source development model, *Journal of Systems and Software* 66 (1) (2003) 65 – 75.
- [22] L. Yu, Understanding component co-evolution with a study on Linux, *Empirical Software Engineering* 12 (2) (2007) 123–141.
- [23] C. Kapsner, M. W. Godfrey, Cloning considered harmful considered harmful: patterns of cloning in software, *Empirical Software Engineering* 13 (6) (2008) 645–692.
- [24] R. Shatnawi, W. Li, The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process, *Journal of Systems and Software* 81 (11) (2008) 1868 – 1882.
- [25] B. Fitzgerald, Has Open Source Software a Future?, in: Feller et al., pp. 93–106.
- [26] A. Fuggetta, Open source software—an evaluation, *Journal of Systems and Software* 66 (1) (2003) 77 – 90.
- [27] T. Østerlie, L. Jaccheri, A Critical Review of Software Engineering Research on Open Source Software Development, in: W. Stanislaw (Ed.), *Proceedings of the 2nd AIS SIGSAND European Symposium on Systems Analysis and Design*, Gdansk University Press, 2007, pp. 12–20.

- [28] A. Capiluppi, P. Lago, M. Morisio, Evidences in the evolution of os projects through changelog analyses, in: Feller et al.
- [29] J. Noll, What Constitutes Open Source? A Study of the Vista Electronic Medical Record Software, in: Boldyreff et al. [51], pp. 310–319.
- [30] I. Stamelos, L. Angelis, A. Oikonomou, G. L. Bleris, Code quality analysis in open source software development, *Information Systems Journal* 12 (1) (2002) 43–60.
- [31] B. Fitzgerald, The Transformation of Open Source Software, *MIS Quarterly* 30 (3) (2006) 587–598.
- [32] The Open Source Initiative - Open Source Licenses (2009).
URL <http://opensource.org/licenses>
- [33] R. Goldman and R. Gabriel, *Innovation Happens Elsewhere*, Morgan Kaufmann, 2005.
- [34] Dirk Riehle, *The Economic Motivation of Open Source Software: Stakeholder Perspectives*, 2007
- [35] E. Raymond, *The Cathedral and the Bazaar*, O'Reilly, 2001
- [36] G. Moore, *Inside the Tornado: Marketing Strategies from Silicon Valley's Cutting Edge*, Harper Perennial, 1999.
- [37] S. Krishnamurthy, "An Analysis of Open Source Business Models," J. Feller et al., eds., *Perspectives on Free and Open Source Software*, MIT Press, 2005, pp. 279-296.
- [38] Brandenburger, A. M., H. Stuart. 1996. Value-based business strategy. *J. Econom. Management Strategy* 5 5–25.
- [39] Brandenburger, A. M., B. J. Nalebuff. 1995. The right game: Use game theory to shape strategy. *Harvard Bus. Rev.* 73 57–71.
- [40] Lakhani, K. R., and E. von Hippel. 2003. How open source software works: “free” user-to-user assistance. *Research Policy* 32:923–943.
- [41] Sandeep Krishnamurthy, *An Analysis of Open Source Business Models*, 2003
- [42] Zoe konovalov, “The economics of open source software”, 2002, pp.35-37
- [43] McKelvey, M. (2001): The Economic Dynamics of Software: Three Competing Business Models Exemplified through Microsoft, Netscape and Linux, *Economics of Innovation and New Technology*, 11: p.127-164.

- [44] Khalak, A. (2000): Economic Model for Impact of Open Source Software, MIT, Department of Mechanical Engineering: Boston.
- [45] Krishnamurthy, S. (2003): An Analysis of Open Source Business Models, in J. Feller, et al. (eds.): Perspectives on Free and Open Source Software. MIT Press: Cambridge, Massachusetts. p. 279-296.
- [46] Onetti A, Verma S. (2008): Licensing and Business Models, report University of Insubria, 2008/6.
- [47] Jesper Holck, Roberto Zicari, and Volker Mahnke “A Framework Analysis of Business Models for Open Source Software”
- [48] Mickos, M. (2006): Personal communications with the authors, November 28, 2006.
- [49] Holck, J., Mahnke, V. and Zicari, R. (2008): Winning Through Incremental Innovation: The Case of MySQL, Proc. IRIS31 – The 31st Information Systems Research Seminar in Scandinavia, Åre, Sweden.
- [50] M. Gruber and J. Henkel, New ventures based on open innovation - an empirical analysis of startup firms in embedded Linux, *International Journal of Technology Management* **33**(4), 256-372 (2006).
- [51] C. Boldyreff, K. Crowston, B. Lundell, A. I. Wasserman (Eds.), Proceedings of the 5th IFIP Working Group 2.13 International Conference on Open Source Systems (OSS2009) – Open Source Ecosystems: Diverse Communities, June 3-6, Skövde, Sweden, Vol. 299/2009 of IFIP International Federation for Information Processing, Springer, Heidelberg, Germany, 2009.
- [52] Blundell R, Bond S. 1998. Initial conditions and moment restrictions in dynamic panel data models. *Journal of Econometrics* **87**: 115-143.

Appendix

Answers should be given for 2004 June

PLEASE NOTE: Data gathered within this project will be used only for scientific research purposes. ELISS II project aims at increasing knowledge on the current situation and evolution of the European software industry.

Data will be handled in compliance with Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on Privacy and Electronic Communications)

For the purposes of the analysis it is very important to distinguish the value ZERO from NON RESPONSES. Please indicate explicitly the value ZERO when needed

GENERAL INFORMATION

Company name _____

Province _____

1. The firm was established in year _____ (please give the year) by how many people? _____ (please give the number)

2. Owners structure. Is the firm owned by other companies?

YES	NO

3. Firm workforce. Please give the number (from **ZERO** to **N**):

Owners	Employees	Freelancers

How many of them develop software? _____ (full time equivalent)

4. Education of firm workforce. Please give the number (from **ZERO** to **N**):

1	Less than high school degree	_____
2	High school degree	_____
3	University students	_____

4	University education (degree)	_____
5	Master	_____
6	Ph.D.	_____

5. Turnover in 2000 in thousand Euros _____

Please note: if the firm was born after 2000, please take turnover of the first year

6. Turnover in 2003 in thousand Euros _____

7. Typology of firms' main customers. Tick only one:

1	Small and medium enterprises	
2	Large firms	
3	University and research centres	
4	Hospitals	
5	Other public bodies	
6	End users	
7	Other, specify _____	

8. How important do you think that patents are in your industry for increasing your revenues?
Tick only one:

1	Very important	
2	Important	
3	Nice to have	
4	Not important	
5	Other, specify _____	

9. With respect to the software industry, you think that patents:

		YES	NO
1	Are important to promote innovation		
2	Humble innovation		
3	Are useful because they create entry barriers		

4	Limit competition		
5	Are useless because software has a short life cycle		
6	Other forms of protection are better for software		
7	Increase innovation costs		
8	Are dangerous because of the sequentiality of the innovation process in the software industry		
9	Foster information disclosure		
10	Other, specify _____		

10. In your opinion, what is the best way to protect innovation in the software industry? _____

DOES THE FIRM SUPPLY OPEN SOURCE BASED – PRODUCTS AND/OR SERVICES AND RELEASE THEM UNDER AN OPEN SOURCE LICENSE?

YES	NO
------------	-----------

IF YES, please indicate the year in which the firm started this activity _____ and then please fill **PART A**

IF NO, has the firm ever do that?

YES	NO
------------	-----------

IF NO, please fill **PART B**

If YES, please indicate the year in which the firm started _____ and gave up this activity _____ and then, please, fill **PART C**

PART A

11. We offer our customers software solutions which are. Tick only one:

Mainly proprietary solutions	Open Source and proprietary software: without no distinctions	Mainly Open Source solutions	Exclusively Open Source solutions

12. Firm's customers. Tick only one:

Prefer Open Source solutions	Have no preference	Prefer proprietary solutions

13. Firm's customers attach importance to the following factors when they choose Open Source software. Pick out the top five factors:

1	Packages that are widely used	
2	Price	
3	After-sales support	
4	Availability of updated releases	
5	Source code availability	
6	Freedom from suppliers	
7	Security	
8	Operating system compatibility	
9	Compatibility with other programs and applications	
10	Total Cost of Ownership (TCO)	
11	Cost for staff training	
12	Reputation of the software producers	
13	Customisation	
14	License characteristics (fees, restrictions, etc.)	
15	Performances	
16	Other, specify _____	

14. Which statement best describes the firm's activities in Open Source software?

		Very important	Nice to have	NOT important
1	We sell Open Source pre-packaged products and include complementary services (e.g. installation, training, etc.)			
2	We adapt pre-existing Open Source programmes and solutions to suit customers' needs			
3	We integrate Open Source programs and modules in new solutions and release them under Open Source licenses			
4	We design and develop on order new solutions for our customers and release them under Open Source licenses			

5	We develop new products from the scratch and put them on the market under Open Source licenses			
6	Other, specify _____			

15. Products offered:

		Proprietary products	Open Source products under copyleft licenses (GPL and GPL-like)	Open Source products under NON copyleft licenses (e. g. BSD, Apache, etc.)
1	NETWORK INFRASTRUCTURE SERVICES – SERVER SIDE (a)			
2	BASIC INTERNET SERVICES – SERVER SIDE (b)			
3	BASIC INTERNET APPLICATIONS – CLIENT SIDE (c)			
4	SOLUTIONS SUITED FOR THE PUBLIC ADMINISTRATION (d)			
5	SOFTWARE FOR NETWORK MANAGEMENT AND MONITORING			
6	Development Tools			
7	Application server			
8	Applications suited for the Scientific Community (e.g. software for data analysis)			
9	Database Management Systems			
10	Content Management Systems			
11	Management Software			
12	E-learning Tools			
13	Workflow systems			
14	E-commerce Solutions			
15	Office Automation Packages			
16	User and Identity Management			
17	Other products, give the number	No. _____	No. _____	No. _____

(a) The category NETWORK INFRASTRUCTURE SERVICES – SERVER SIDE includes: File Server, Print Server, Backup Systems

(b) The category BASIC INTERNET SERVICES – SERVER SIDE includes: Mail Server, Proxy Server, Web Server

(c) The category BASIC INTERNET APPLICATIONS – CLIENT SIDE includes: E-mail Client, Instant Messaging, Web Browser

(d) The category SOLUTIONS SUITED FOR THE PUBLIC ADMINISTRATION includes: Public Administration Protocol Management and Certified e-mail. Please note: if the firm supplies Mail Servers to public bodies must not refer to this category. A mail server, in fact, is a general purpose product

(e) The category SOFTWARE FOR NETWORK MANAGEMENT AND MONITORING includes: Firewall, Antispam, Antivirus, Digital Signature Systems

16. Services offered:

		For proprietary products	For Open Source products distributed under copyleft licenses (GPL and GPL-like)	For Open Source products distributed under NON copyleft licenses (e.g. BSD, Apache, etc.)
1	Consultancy			
2	Integration			
3	Installation			
4	Assistance			
5	Maintenance			
6	System Management			
7	Training			
8	Application Management			
9	Adapting codes written by third parties to suit customers' needs			
10	On order software development from the scratch			
11	Generating documentation			
12	Other services, give the number	No. _____	No. _____	No. _____

17. What kind of operating systems does the firm use?

		YES	NO
1	Microsoft operating systems (e.g. Windows 2000, NT, XP, etc.)		
2	Unix-like operating systems		
3	Linux operating systems (e.g. Red Hat, SuSe, Mandrake, etc.)		
4	Others, specify _____		

18. What percentage of turnover is generated by Open Source software? Give an exact figure or choose one of the following:

		Year 2000	Year 2003
1	Exact figure	_____	_____
2	0%		
3	Less than 10%		

4	Between 10% and 30%		
5	Between 31% and 50%		
6	Between 51% and 70%		
7	Between 71% and 90%		
8	Between 91 and 99%		
9	100%		

Please note: if sales of Open Source began after 2000, please take Open Source turnover of the first year

19. Pick out the top three obstacles to Open Source diffusion:

1	Firms' customers need to adapt their organisational processes to the Open Source software	
2	The firm needs to adapt its organisational processes to the Open Source software	
3	Open Source software still not being widely used	
4	The widespread use of incompatible proprietary operating systems	
5	The widespread use of incompatible proprietary applications	
6	Open Source software is not secure because of source code accessibility	
7	The possibility of patenting software	
8	Open Source products are more difficult to use than their proprietary equivalents	
9	Open Source products have a worse after-sales service than their proprietary equivalents	
10	Commercial strategies of the proprietary software firms for stemming Open Source software diffusion	
11	Copyleft license schemes' characteristics	
12	Lack of personnel skilled in Open Source software to be hired by the firm	
13	Other, specify _____	

20. Pick out the top three incentives that have led the firm to offer Open Source solutions to its customers:

1	Being independent from the price and licensing policies of large software producers	
2	Exploiting the possibility Open Source software offers to be innovative while staying small	
3	Satisfying private customers' demands for Open Source Software	
4	Addressing Public Administration customers, who are moving towards the new open standards	
5	Getting feedback and contributions from the Open Source community	
6	Lowering development costs exploiting the existing Open Source code for new software solutions	
7	Gaining access to products which are not available on the proprietary software market	
8	Having better pricing option	
9	Fostering the diffusion of the Open Source software	
10	Make partnerships with other software firms that work with Open Source software	
11	Enter markets that otherwise would be out of reach	
12	Making clear our own innovative contributions through source code accessibility	
13	Thinking that the demand for Open Source software is going boom very soon. We want to be ready for that	
13	Other, specify _____	

21. Take into account your typical market (including all the customers, not only the firm's ones). If you were asked to quote the percentage of customers that adopted Open Source solutions in 2003, what estimation would you make? And for 2006?

	2003	2006
Client side	_____ %	_____ %
Server side	_____ %	_____ %

RELATIONSHIPS WITH THE OPEN SOURCE COMMUNITY

An Open Source project is a software development project showing the following features

- The code is freely available on the Internet
- The code is released under an Open Source license
- Every one is allowed to take part in the project
- Collaboration among developers shape the software production mode

Taking part in an Open Source project means giving concrete contributions to software development and improvement. For instance

- Downloading the code from the Internet, installing the software and providing feedback on how it runs
- Writing documentation (and/or translating the existing documentation into another language)
- Finding bugs
- Fixing bugs
- Contributing code
- Providing user assistance within the project's mailing lists

Coordinating an Open Source project means managing all the activities dealing with software development

- Defining project's goals
- Releasing new versions very often
- Motivate the community to provide contributions
- Settling conflicts among developers
- Avoid forking

22. Give details of Open Source projects:

	From the very start of the Open Source activity	During 2003
The firm has participated to (on behalf of itself)	No. of projects___	No. of projects___
The firm has coordinated (on behalf of itself)	No. of projects___	No. of projects___

23. During 2003, the firm took part in Open Source projects:

		No. of projects	Man/hours
1	Writing documentation	_____	_____
2	Fixing bugs	_____	_____
3	Providing user assistance within the projects' mailing lists	_____	_____
4	Contributing code that was accepted in projects' official versions	_____	_____
5	Contributing code that was NOT accepted in projects' official versions	_____	_____
6	Other, specify _____	_____	_____

24. What kind of feedback and contributions does the firm receive from the Open Source community?

		YES	NO
1	Documentation	_____	_____
2	Bug fixing	_____	_____
3	Customer assistance within the project's mailing lists	_____	_____
4	Code	_____	_____
5	Other, specify _____	_____	_____

25. Write down the top three Open Source projects in which the firm takes part:

1	_____
2	_____
3	_____

26. The firm has authorised its employees to contribute during working hours to Open Source projects that are not directly related to its own?

YES	NO
_____	_____

27. If YES, how much time do they spend on this activity on average?

1	Less than 10% of the working hours	_____
2	Between 10% and 20% of the working hours	_____

3	More than 20% of the working hours	
---	------------------------------------	--

28. Give a mark from 1 (= not at all important) to 5 (= very important) for each statement. With reference to the Open Source projects in which you are currently engaged, how important are:

		1	2	3	4	5
1	The outcome of the project for our business					
2	Our participation for motivating developers to contribute					
3	Our contribution for the overall success of the project					
4	Other, specify _____					

29. The firm participates in Open Source projects on behalf of itself or allows its employees to participate because of the following motivations. Pick out the top three motivations:

1	The whole society takes advantage of the development of Open Source software	
2	We agree with the values of the Open Source movement	
3	Open Source programming is a creative activity	
4	Being acknowledged as active members of the community of Open Source developers	
5	Increasing the amount of code at the disposal of the community of Open Source developers	
6	Getting commercial visibility	
7	Getting more quickly suggestions and contributions from other programmers on programs in our interest	
8	Directing projects towards themes in our interest	
9	Taking part in Open Source projects prompts our ability to innovate	
10'	Taking part in Open Source projects allows our developers to increase their competences	
11	All the innovative companies take part in Open Source projects	
12	Other, specify _____	

PART B

30. Products offered:

1	NETWORK INFRASTRUCTURE SERVICES – SERVER SIDE (a)	
2	BASIC INTERNET SERVICES – SERVER SIDE (b)	
3	BASIC INTERNET APPLICATIONS – CLIENT SIDE (c)	
4	SOLUTIONS SUITED FOR THE PUBLIC ADMINISTRATION (d)	
5	SOFTWARE FOR NETWORK MANAGEMENT AND MONITORING	

6	Application server	
7	Applications suited for the Scientific Community (e.g. software for data analysis)	
8	Content Management Software	
9	Database Management System	
10	Development Tools	
11	E-commerce Solutions	
12	E-learning Tools	
13	Management Software	
14	Office Automation Packages	
15	User Management and Identity Management	
16	Workflow systems	
17	Other products, give the number	No. _____

(a) The category NETWORK INFRASTRUCTURE SERVICES – SERVER SIDE includes: File Server, Print Server, Backup Systems

(b) The category BASIC INTERNET SERVICES – SERVER SIDE includes: Mail Server, Proxy Server, Web Server

(c) The category BASIC INTERNET APPLICATIONS – CLIENT SIDE includes: E-mail Client, Instant Messaging, Web Browser

(d) The category SOLUTIONS SUITED FOR THE PUBLIC ADMINISTRATION includes: Public Administration Protocol Management and Certified E-mail.
Please note: if the firm supplies Mail Servers to public bodies must not refer to this category. A mail server, in fact, is a general purpose product

(e) The category SOFTWARE FOR NETWORK MANAGEMENT AND MONITORING includes: Firewall, Antispam, Antivirus, Digital Signature Systems

31. Services offered:

1	Consultancy	
2	Integration	
3	Installation	
4	Assistance	
5	Maintenance	
6	System Management	
7	Training	
8	Application Management	
9	Adapting codes written by third parties to suit customers' needs	
10	On order software development from the scratch	
11	Generating documentation	

12	Other services, give the number	No. _____
----	---------------------------------	-----------

32. What kind of operating systems does the firm use?

		YES	NO
1	Microsoft operating systems (e.g. Windows 2000, NT, XP, etc.)		
2	Unix-like operating systems		
3	Linux operating systems (e.g. Red Hat, SuSe, Mandrake, etc.)		
4	Others, specify _____		

33. Take into account your typical market (including all the customers, not only the firm's ones). If you were asked to quote the percentage of customers that adopted Open Source solutions in 2003, what estimation would you make? And for 2006?

	2003	2006
Client side	_____%	_____%
Server side	_____%	_____%

34. Pick out the top three motivations because the firm DOES NOT offer Open Source based products and services to its customers:

1	We have no demand for this kind of software	
2	Personnel training characteristics	
3	The competences for working with this kind of software are not available	
4	We do not want to work with Open Source licenses	
5	Open Source software does not allow to make profits	
6	The lack of a strong IPRs regime	
7	Compatibility issues of Open Source software	
8	Security concerns	
9	The firm needs to adapt its organisational processes to the Open Source software	
10	Other, specify _____	

35. Firm's customers attach importance to the following factors when they choose software. Pick out the top five factors:

1	Packages that are widely used	
2	Price	
3	After-sales support	
4	Availability of updated releases	
5	Source code availability	
6	Freedom from suppliers	
7	Security	
8	Operating system compatibility	
9	Software compatibility	
10	Total Cost of Ownership (TCO)	
11	Cost for staff training	
12	Reputation of the software producers	
13	Customisation	
14	License characteristics (fees, restrictions, etc.)	
15	Performances	
16	Other, specify _____	

36. Pick out the top three obstacles to Open Source diffusion:

1	Firms' customers need to adapt organisational processes to the Open Source software	
2	Open Source software still not being widely used	
3	The widespread use of incompatible proprietary operating systems	
4	The widespread use of incompatible proprietary applications	
5	Open Source software is not secure because of source code accessibility	
6	The possibility of patenting software	
7	Open Source products are more difficult to use than their proprietary equivalents	
8	Open Source products have a worse after-sales service than their proprietary equivalents	
9	Commercial strategies of the proprietary software firms for stemming Open Source software diffusion	

10	Copyleft license schemes' characteristics	
11	Lack of personnel skilled in Open Source software to be hired by our company	
12	Other, specify _____	

PART C

37. Products offered:

1	NETWORK INFRASTRUCTURE SERVICES – SERVER SIDE (a)	
2	BASIC INTERNET SERVICES – SERVER SIDE (b)	
3	BASIC INTERNET APPLICATIONS – CLIENT SIDE (c)	
4	SOLUTIONS SUITED FOR THE PUBLIC ADMINISTRATION (d)	
5	SOFTWARE FOR NETWORK MANAGEMENT AND MONITORING	
6	Application server	
7	Applications suited for the Scientific Community (e.g. software for data analysis)	
8	Content Management Software	
9	Database Management System	
10	Development Tools	
11	E-commerce Solutions	
12	E-learning Tools	
13	Management Software	
14	Office Automation Packages	
15	User Management and Identity Management	
16	Workflow systems	
17	Other products, give the number	No. _____

(a) The category NETWORK INFRASTRUCTURE SERVICES – SERVER SIDE includes: File Server, Print Server, Backup Systems

(b) The category BASIC INTERNET SERVICES – SERVER SIDE includes: Mail Server, Proxy Server, Web Server

(c) The category BASIC INTERNET APPLICATIONS – CLIENT SIDE includes: E-mail Client, Instant Messaging, Web Browser

(d) The category SOLUTIONS SUITED FOR THE PUBLIC ADMINISTRATION includes: Public Administration Protocol Management and Certified E-mail.
Please note: if the firm supplies Mail Servers to public bodies must not refer to this category. A mail server, in fact, is a general purpose product

(e) The category SOFTWARE FOR NETWORK MANAGEMENT AND MONITORING includes: Firewall, Antispam, Antivirus, Digital Signature Systems

38. Services offered:

1	Consultancy	
2	Integration	
3	Installation	
4	Assistance	
5	Maintenance	
6	System Management	
7	Training	
8	Application Management	
9	Adapting codes written by third parties to suit customers' needs	
10	On order software development from the scratch	
11	Generating documentation	
12	Other services, give the number	No. _____

39. What kind of operating systems does the firm use?

		YES	NO
1	Microsoft operating systems (e.g. Windows 2000, NT, XP, etc.)		
2	Unix-like operating systems		
3	Linux operating systems (e.g. Red Hat, SuSe, Mandrake, etc.)		
4	Others, specify _____		

40. Take into account your typical market (including all the customers, not only the firm's ones). If you were asked to quote the percentage of customers that adopted Open Source solutions in 2003, what estimation would you make? And for 2006?

	2003	2006
Client side	_____ %	_____ %
Server side	_____ %	_____ %

41. Pick out the top three motivations because the firm GAVE UP the offering of Open Source based products and services to its customers:

1	We have no demand for this kind of software	
2	Personnel training characteristics	
3	The competences for working with this kind of software are not available	
4	We do not want to work with Open Source licenses	
5	Open Source software does not allow to make profits	
6	The lack of a strong IPRs regime	
7	Compatibility issues of Open Source software	
8	Security concerns	
9	The firm needs to adapt its organisational processes to the Open Source software	
10	Other, specify _____	

42. Firm's customers attach importance to the following factors when they choose software. Pick out the top five factors:

1	Packages that are widely used	
2	Price	
3	After-sales support	
4	Availability of updated releases	
5	Source code availability	
6	Freedom for suppliers	
7	Security	
8	Operating system compatibility	
9	Software compatibility	
10	Total Cost of Ownership (TCO)	
11	Cost for staff training	
12	Reputation of the software producers	
13	Customisation	
14	License characteristics (fees, restrictions, etc.)	
15	Performances	

16	Other, specify _____	
----	----------------------	--

43. Pick out the top three obstacles to Open Source diffusion:

1	Firms' customers need to adapt organisational processes to the Open Source software	
2	Open Source software still not being widely used	
3	The widespread use of incompatible proprietary operating systems	
4	The widespread use of incompatible proprietary applications	
5	Open Source software is not secure because of source code accessibility	
6	The possibility of patenting software	
7	Open Source products are more difficult to use than their proprietary equivalents	
8	Open Source products have a worse after-sales service than their proprietary equivalents	
9	Commercial strategies of the proprietary software firms for stemming Open Source software diffusion	
10	Copyleft license schemes' characteristics	
11	Lack of personnel skilled in Open Source software to be hired by our company	
12	Other, specify _____	