# Economic aspects and business models of free software

**Amadeu Albós Raya**
**Lluís Bru Martínez**
**Irene Fernández Monsalve**
**Edited by David Megías Jiménez**

# Economic aspects and business models of free software

by Amadeu Albós Raya, Lluís Bru Martínez, Irene Fernández Monsalve, and David Megías Jiménez

Publication date 2010-02-01

GNUFDL

# Table of Contents

# List of Examples

# preliminars

## info

First edition: September 2010
© Amadeu Albós Raya, Lluís Bru Martínez, Irene Fernández Monsalve
All rights reserved
© of this edition, FUOC, 2010
Av. Tibidabo, 39-43, 08035 Barcelona
Design: Manel Andreu
Publishing: Eureca Media, SL
ISBN: 978-84-691-8694-7
Legal deposit: B-2.239-2009

# introduction

In our modern society, the importance of the information and communications technology (ICT) sectors, i.e. telecommunications, computing and electronics (production and consumer), has reached unprecedented heights. These industries are innovating at a dramatic pace and converging into one, handling, transmitting and reproducing information with the use of common procedures: digital technologies.

However, far more important than the changes that have taken place in the industries directly related to ICTs are the changes and improvements affecting other business activities, particularly the more traditional ones, where procedures have undergone a major overhaul. The effects on the economic fabric have led to the emergence of new business areas in order to meet new market demands that were previously unheard of.

Nonetheless, no economic legislation has changed and none of the economic phenomena related to ICTs are qualitatively new. What has changed, if anything, is the relative importance of certain economic effects on our society, such as intellectual property policies and product compatibility. If we look more specifically at the economic effects on the software industry, we will reach a similar conclusion.

Recently, in the software industry, we have seen how free software has been opened up and deployed among the general public. Free software has positioned itself in diverse sectors of the generic software market as a valid and viable alternative, and conclusively so in some specialised sectors (such as web server software). This situation has affected the proprietary-based software industry and led to the rise of new business models related to the dissemination, development, support and implementation of free software.

### Important

The aim of the *Economic aspects and free software business models* subject is to provide the necessary knowledge to understand and implement free software economics through the study of economic aspects and analysis of the related business models, in combination with the opportunities offered by this new market.

The first module of the subject outlines the economic concepts required to analyse the economic structure of the software industry in general and to conform a specific business model based on free software.

The second module of the subject briefly describes the main features of the software market in relation to the consolidated business models and potential customers of companies based on free software.

The third module of the subject introduces software from a business angle, focusing on the market in which the different solutions compete and on the business strategy for attracting potential customers.

The fourth module explores the business models related to free software. To begin with, it analyses the classifications proposed by some authors of note. It then defines a particular classification for the subject that takes into account the most relevant financial characteristics of the business.

The fifth module examines free software development from a company perspective. Here, we will consider both the most relevant features of the software project and the special features relating to the community of free software users and their functional and legal management.

The sixth module describes the main advantages and disadvantages of the free software model for the business strategy, considering both the perspective of the customer and of the company itself and the business model it adopts.

The seventh and final module of the subject analyses free software as an economic model, focusing both on the fundamental bases of its existence and on present and future implications in relation to the free software business.

# objectives

After completing this subject, students should have achieved the following aims:

1. To identify and understand the most relevant economic factors affecting the technology industry and the free software industry in particular.
2. To understand the different business models that can be exploited in the software industry, along with their key features.
3. To obtain a detailed knowledge of the different business models related to free software and their business opportunities.
4. To understand and relate the economic, technical and legal factors needed to create and sustain a business based on free software.
5. To analyse cases of free software exploitation in the private sector and their relationship with the technology market.
6. To obtain a detailed knowledge of the strategies for developing and exploiting free software for economic gain.
7. To plan and design sound and feasible businesses based on the exploitation of free software.

# Resources

**Associations of free software companies:**

- Asociación Catalana de Empresas por el Software Libre: (www.catpl.org [http://www.catpl.org]).
- Asociación de Empresas de Software Libre de Canarias (www.eslic.info [http://www.eslic.info]).
- Asociación de Empresas de Software Libre de Euskadi (www.esle-elkartea.org [http://www.esle-elkartea.org]).
- Asociación de Empresas Gallegas de Software Libre (www.agasol.org [http://www.agasol.org]).
- Asociación Madrileña de Empresas de Software Libre (www.solimadrid.org [http://www.solimadrid.org]).

- Open Source Business Organisations of Europe (www.obooe.eu [http://www.obooe.eu]).

**Setting up a business:**

- Centro Europeo de Empresas e Innovación del Principado de Asturias: Guía para la creación de empresas y elaboración del Plan de Empresa (http://www.guia.ceei.es).

**Conferences (free software and business):**

- Open Source Business Conference (OSBC) (http://www.osbc.com/live/13/events/13SFO07A/SN441958/CC141932/ and http://www.osbc.com).

- WhyFLOSS (http://www.whyfloss.com/es/conference).

**Corporate data:**

- Spanish Chamber of Commerce (http://www.camerdata.es).

- Hoovers (http://www.hoovers.com).

**Case studies:**

- Avanzada7: "Business models based on Asterisk: The case of Avanzada7" (http://www.whyfloss.com/es/conference/madrid08/getpdf/64).

- IBM migration to Free Desktops (http://www.channelregister.co.uk/2007/02/13/ibm_open_client).

- Liferay: "Liferay Enterprise Portal: The project, the product, the community and how to extend it" (http://www.whyfloss.com/es/conference/madrid08/getpdf/66).

- Openbravo: "Openbravo: keys to success in free software application development" (http://www.whyfloss.com/es/conference/madrid08/getpdf/49).

- Red Hat and JBoss: "Is Open Source viable in Industry? The case of Red Hat and JBoss" (http://www.whyfloss.com/es/conference/madrid08/getpdf/68).

- Various cases (www.opensourceacademy.gov.uk/solutions/casestudies [http://www.opensourceacademy.gov.uk/solutions/casestudies]).

**Blogs on free software, innovation and business**

**Galopini, R.** *Commercial open source software* <http://robertogaloppini.net/> [Consulted in March 2009]

**Oswalder, A.** *Business model design and innovation blog.* <http://business-model-design.blogspot.com/> [Consulted in March 2009]

**The 451 group.** *451 caos theory: A blog for the enterprise open source community.* <http://blogs.the451group.com/opensource/> [Consulted in March 2009]

# Chapter 1. Basic notions of economics

Lluís Bru Martínez

GNUFDL
2010-02-01

# preface

This first module introduces the main concepts of product economics and focuses particularly on the specific features of the business of information and communication technologies. These concepts are intended to lay the foundations for understanding the different actions and business models established by the business policy, which we will see later.

The first section introduces the basic notions of product value according to supply and demand and of competitive advantage over rivals as essential tools for business viability.

In the second section, we will describe the main economic effects relating to the features of technology products and software in general. In it, we will explain how a company can act on the market by establishing a policy to manipulate these effects in order to create a scenario that will afford it the best possible position over its competitors.

# objectives

After completing this module, students should have achieved the following aims:

1. To understand the basics of the relationship between supply and demand, particularly with concepts concerning value creation.
2. To identify and analyse the key economic features of the software industry.
3. To obtain a detailed knowledge of and link the economic effects associated with the software market.
4. To identify and analyse the economic effects likely to transmit value or a competitive advantage to products based on free software.
5. To obtain a detailed understanding of the management policies and strategies of the free software market.

# Value creation

To ensure the viability of a given business, there must be people or businesses willing to pay, as customers, for the product or service offered to them, and these payments must compensate their providers for the expenses incurred. First of all, we will explain in simple terms the basic economic concepts at work in this interaction between the company organising the business and the prospective clients of its product or service.

## Product demand

First of all, we need to introduce some of the possible rules of conduct for the businesses and households that we want to convert into customers of our business.

A consumer (if we are talking about consumer goods) or a company (if purchasing machinery, raw materials, etc). will consider buying a particular product or service if the amount of money asked of them in exchange (payment) seems reasonable to them.

In this situation, the prospective buyer makes the following argument:

1. Firstly, he/she considers it reasonable to pay at most an amount of money V to acquire the product or service being offered in exchange. Therefore, if he/she is asked for an amount of money P less than V, he/she will consider it worthwhile to acquire the product. So, for someone to consider becoming our customer, the following conditions must be fulfilled:

   Assessment of product – its price = V-P > 0

   To put it another way, a company will not be paid more than V for its product or service. However, this will not guarantee that the customer will buy the product.

2. Secondly, the customer will compare this offer with the available alternatives. Of two or more similar products, the consumer will choose the one in which the difference of V–P is greater.

### Example 1.1. Example

A family is thinking about buying a car. The family values the model of manufacturer A at €40,000 (Va = €40,000) although the selling price is €30,000, Pa = 30,000. The family values the model of manufacturer B at a lower price; to be exact, let's suppose that it values the car less due to inferior features (for example, it is a smaller vehicle) at €35,000, Vb = €35,000.

The family in our example will buy the model of manufacturer A, even though it is more expensive, so long as the car of manufacturer B is sold at over €25,000, and vice versa: it will buy manufacturer B's car if it is cheap enough, i.e. if its price is under €25,000:

We can conclude that:

It will only purchase the product of manufacturer A if

Va–Pa = 40,000 – 30,000 > Vb–Pb = 35,000–Pb,

i.e. only if Pb > 25,000.

It will only purchase the product of manufacturer B if

Va–Pa = 40,000 – 30,000 < Vb–Pb = 35,000–Pb,

i.e. only if Pb < 25,000.

## Important

The**demand** for a particular product consists of the series of customers obtained for each possible price of the product in question.

In our example, if every family values these products in the same way for prices over €25,000, there will be no demand for manufacturer B's product, while for lower prices, we have the demand of all of the families that value the product of the same family that we have discussed.

On what does the value V that a prospective customer gives to a product or service depend? First and foremost, it depends on the intrinsic ability of the product to meet the customer's needs, but also:

**1)**On the customer's ability to adequately evaluate the product, which depends largely on his/her background and education.

It would be difficult for a customer to evaluate the GNU/Linux operating system, for example, if he/she does not even know what an operating system is and has never even considered that a computer is not necessarily required to have the Microsoft Windows operating system installed.

**2)**On the importance of the availability of secondary products to complement the main product that we are being offered (a car is more valuable if roads are better and petrol stations are easy to come across, and less valuable if roads are congested, public transport is good, if petrol becomes more expensive, etc).

**3)**On the real spread of the product offered to us, i.e. the number of other people who have it: telephone and e-mail are more valuable the more people who use them.

# Product supply

For their part, employers will concentrate on a certain product so long as they can obtain a reasonable profit from it, which requires them to consider two key aspects:

1. The costs of looking after customers.

2. What they would gain by engaging in another activity.

### Example 1.2. Example

Let's say a couple decides to open a bar. At the end of the first year, they have obtained a revenue of €150,000, while the costs of serving patrons, hire of the premises, etc. amount to €120,000. We can see that the first condition is fulfilled because the revenue has far exceeded the costs; accountants would tell us that we have a profit, since the revenue covers costs.

However, imagine that, in order to open the bar, this couple gave up their jobs as paid workers, which had given them an annual income of €40,000. These alternative incomes are what economists call the opportunity cost of setting up the bar as a business. We can see that the second aspect is not covered in the example:

The business is not really making money because

Revenue – costs = 150,000 – 120,000 = 30,000

< Opportunity cost = 40,000

Of course, this couple may still prefer to run the bar than to work as employees, so we can consider their sacrifice in terms of the money left over at the end of the year reasonable if the satisfaction of running their own business compensates for this. Our point is, firstly, that they are not running a good business from a strictly monetary perspective, and secondly, that their decision will only seem reasonable if it is a lucid decision, that is, if they consciously accept this loss of revenue; it would not be reasonable if they did not accept that they would have less money.

To turn the bar into a good business, the profits must outweigh the "profit" from the alternative activity or opportunity cost.

If the annual revenue of the bar is €180,000, for example, then we would be talking about a good business:

This would be a good business because

Revenue – costs = 180,000 – 120,000 = 60,000

> Opportunity cost = 40,000

We can conclude that, in order to sustain a business, the profits obtained must exceed the opportunity costs of engaging in alternative activities.

# Value creation and competitive advantage

Based on what we have seen so far, we can consider the requirements that need to be met for a business to be profitable.

Firstly, it is necessary to create value, i.e. that the valuation V made of the product on offer by prospective customers exceeds its costs:

## Important

For a business to be viable, it is essential for

product valuation – costs – opportunity cost > 0

Only when this is true can we say that a company creates value and can be viable, because only then can we find a price that is fair for both the customer and the company.

### Example 1.3. Example

If the value of a product for a customer is V = €100 and the cost of taking care of the latter is C = 60, we can find a satisfactory price for the customer and the company, such as P = 80, and the exchange will be satisfactory for both because the following conditions hold true:

V – P > 0

and

P – total costs > 0

Fulfilment of the condition V – C > 0, however, does not guarantee the viability of a business. To illustrate this, we will go back to our previous example in section 1.1 (product demand), though this time from the point of view of two rival companies trying to win over a customer:

### Example 1.4. Example

We have two car manufacturers offering two similar models. We have seen that the family valued one of the cars at Va = 40,000 and the other at Vb = 35,000.

Imagine that the manufacturing costs of company A are Ca = 20,000, while those of company B are Cb = 10,000. Both companies manufacture at costs far below the respective Va and Vb valuations.

Therefore, if they had no rival, both companies would clearly be viable as a business.

Now imagine that company B decides to sell its vehicles at the price of Pb = 18,000. Customer satisfaction is

Vb – Pb = 35,000 – 18,000 = 17,000.

Company A must provide a greater – or at least similar – level of satisfaction to gain customers:

Company A gains customers if:

Va – Pa > Vb – Pb = 17,000 only if < 23,000.

Company A therefore has the ability to attract clients and cover costs. However, if we look closely, we see that this company is at the mercy of its rival:

If company B decides to lower its prices to less than 15,000 (e.g. Pb = 14,000), company A cannot continue to attract customers without incurring losses:

Vb – Pb = 35,000 – 14,000 = 21,000 and

Va – Pa > Vb – Pb = 21,000 only if Pa < 19,000, but then

Pa – Ca < 0 !

In this example, company B has a competitive advantage over its rival, company A. The result is that one of two situations occurs:

1) Company B attracts all of the customers, such as when it establishes Pb = 14,000, or

2) The two companies share out the customers, but company B makes more money on each:

They divide the customers between them if Va – Pa = Vb – Pb, but this means that Pa – Ca < Pb – Cb, for example if Pb = 18,000 and Pa = 23,000.

## Important

Ultimately, the company with the competitive advantage will guarantee its survival and, in all events, make more money than its rivals.

In the above example, company B had a competitive advantage in costs: although the product was perhaps not best suited to the needs of customers, Va > Vb, was able to produce a reasonable product with costs well below those of its rival.

### Example 1.5. Inditex

An interesting example for us to consider on this course is **Inditex**, the company that owns the clothing retailer Zara. The fashion clothing industry, of which the company forms part, is a highly competitive sector in which companies can copy each other's designs without limits. Nevertheless, there is a very high degree of inventiveness, with new models appearing every season, year after year (and naturally, a considerable number of companies that engage in this activity), and at very low prices. As customers, therefore, we can reap the benefits of a highly competitive and innovative industry.

Despite all of this, Inditex manages to expand its market share each year (i.e. it attracts an increasing proportion of customers) because of its competitive advantage in costs, which appears to consist basically of (1) rapidly detecting the designs that sell best in a given season and (2) immediately adapting production to these designs. As a result, costs are lower because it does not produce clothing that does not sell and it sells a lot of the clothing preferred that year.

And it would appear that this achievement is no mean feat, because its competitors are incapable of copying their behaviour (at least in such a clever way).

#### Important

A company with a competitive advantage in costs will gain more customers and obtain higher profits because it can sell its products more cheaply.

Alternatively, a company might have a competitive advantage through differentiation, i.e. in offering a product more highly valued than that of its rivals at a reasonable cost.

### Example 1.6. Adobe

Adobe and its Acrobat software is a good example of a better valued product at a reasonable cost.

And this superior valuation can be general, in the sense that all potential customers consider the product to be of a higher quality (this is the case of prestigious German car brands, for example), or of a niche, i.e. it is a specialised product for a particular type of customer (any village shop fulfils this requirement: it is a shop geared to a particular type of customer, namely, the residents of the village, the only ones for whom it is more convenient to buy bread or the newspaper there).

#### Important

Competitive advantage through differentiation allows the company to sell more expensively without losing customers.

## Summary

We have seen in basic terms and from an economic point of view what setting up a **viable business** is all about. To summarise, it consists of creating a product or service that is beneficial to our customers, so that we can charge for it while keeping costs under control.

When setting up a business based on free software, the crucial financial question is: what product or service can I charge the customer for? Before moving on to discuss this in the next section, we will look at a series of relevant economic features of the software industry that we will need to understand in order to answer this question.

# Economic features of the software industry

As we explained at the beginning, no economic legislation has changed and none of the economic phenomena related to information and knowledge technology industries are qualitatively new. What has

changed, if anything, is the relative importance of certain economic effects on our society. In ICT industries specifically, in the market interaction between companies and their customers, there is a series of very important economic phenomena that can distort the operation of these markets. We will now look briefly at the following effects:

1) The costs of copying and distributing digital technology.

2) The economics of intellectual property and ideas.

3) Complementarities.

4) Network effects.

5) Compatible products and standards.

6) Costs of change and captive customers.

7) Policies of compatibility and standardisation within a platform and between platforms.

A recent example of this last point is **compatibility across platforms and the policy** adopted on this issue by the **proprietary software company Microsoft**, which has led to the intervention of the European Commission in defence of **free competition** between companies. Given its importance for the proper conduct of business models based on free software, we will also briefly discuss the approach of the European Commission to the matter.

# The costs of producing, copying and distributing digital technology

Digital technology has a very specific cost structure: it is very expensive to develop a specific product as this requires major investments, and we cannot simply half-develop it.

However, making high quality copies of the developed product and distributing them is relatively cheap.

## Important

Thus, it is very economical to serve additional customers; the expensive part is the initial investment that will lead to the development of a product around which we can organise a business.

## Commercial aviation

Similarly, a commercial aviation company must make a big investment in an aircraft if it wants to set up frequent connections between two airports. It is no use trying to purchase half an airplane, the company will need to buy the whole aircraft. However, serving additional customers – until the plane is full – will work out very cheap for the company.

Naturally, the huge reduction in the costs of copying and distributing the products and services developed with digital technology has led to significant changes in certain industries.

### Example 1.7. The music industry

A typical example is the music industry, which was based around control over the copying (understood to mean a copy of a similar quality; with analogue technology, the sound quality of a cassette tape copy was far inferior to that of a record or CD) and distribution of the product (primarily through specialised shops).

# The economics of intellectual property and ideas

## Important

ICTs are characterised by the fact that they allow the manipulation, broadcasting and reproduction of information and ideas. As a result, the advance of these technologies has the basic effect of **encouraging the spread of ideas and their use**.

Ideas, as an economic asset, have the quality of being **non-rival goods:** just because a person uses an idea does not mean that others cannot use it too.

### Example 1.8. Non-rival goods

If Peter eats an apple, John cannot eat it. In contrast, if Peter uses a recipe, John can also use it.

ICT industries spend a lot of financial resources on **developing new knowledge**, with the aim of making a profit on the exploitation of these ideas. From the point of view of the interest of general society, every time new knowledge arises, whether it is a scientific discovery, a new technique, or something else, the diffusion of this new idea poses a problem. Firstly, it is clear that once this new knowledge is available, it is in the interest of society to disseminate the idea as far as possible. However, the companies that have developed this knowledge have done so in order to gain a profit from it, and they can only do this by restricting access to the new knowledge. Without some form of **protection against the immediate dissemination** of this knowledge, we run the risk that companies will not invest money in the search for and development of new ideas and knowledge.

Advanced societies have created different institutions and mechanisms to facilitate the generation of new scientific and technical knowledge. Scientific creation is financed through public resources. The development and funding of more practical and applied knowledge for the creation of new production techniques and new products is generally left to the private sector. In these cases, public institutions adopt the role of promoting private-sector activity by protecting intellectual property through the institution of a series of legal concepts, most notably **copyright**, **patents**and **trade secrets**.

## Important

*Copyright*protects the particular expression of an idea.

### Example 1.9. Cases of copyright

A typical example is the right of the author of a song or book over his or her work, which means that nobody can publish or distribute it without his/her consent. A person or company that makes a useful discovery may apply for a patent on it, which prohibits others from using this discovery without consent for a specified number of years (usually 20). Lastly, with trade secrets, companies can keep new knowledge secret and receive legal protection for theft. In this case, the inventor is obviously not protected if others make the same discoveries independently through their own efforts.

While proper use of some of these concepts of intellectual property protection may actually stimulate technical and economic progress, unfortunately, they pose two problems: it is highly questionable that all these legal concepts really do protect the development of ideas and that, in recent years, many companies have made spurious use of the legal concepts that could be useful for them. Instead of legitimately protecting their innovation, many companies use their copyrights and patents as anti-competitive instruments to safeguard their market power and make it harder for more innovative rivals to enter.

In the case of software, the emergence of proprietary systems has made it easy for companies to keep trade secrets due to the possibility of distinguishing between the software's source code and binary code. We

can use a program, i.e. we can get the hardware – be it a computer, mobile phone, game console, ATM, etc. – to work with a computer program by incorporating the binary code on to the computer without having access to its source code. Therefore, proprietary software companies use a business model based on charging money for providing a copy of the binary code of their software. The result is that, without knowing the source code, we cannot discover why the program works one way but not in another, and we naturally cannot edit it to allow us to do other things.

The **trade secret** (not revealing the source code), then, allows companies firstly to hide the developed product from their rivals and then, despite everything, to sell a product to consumers (the binary code of the software program).

## Important

**Free software** is the exact opposite since it is based on **sharing the source code** of the program. As we shall see, this requires the development of an entirely different business model based on offering a service: the ability to modify and adapt the software to customer needs using the expertise and knowledge of the computer engineer.

**Copyright, patents and innovation**

P. And you don't agree with patents in software either...

R. Let's just say that I am very sceptical that they serve the purpose they were supposedly designed for. Software is an industry where innovation is sequential. Every new discovery or improvement is constructed on what has been developed before, like a tower. A patent applied at a certain level of the tower slows down further developments. In practice, this works like a monopoly.

Interview with Eric Maskin, 2007 Nobel Laureate in Economics, published in *El País*, 29/06/2008.

### Recommended reading

*El País*. You can read the full interview in the article published in*El País* on 29/06/08 "Es difícil prevenir una burbuja"

Is it true that a creator is really that unprotected without copyright or patents on their ideas? Many creators seem to think so. For example, in a discussion with the CEO of the Bimbo company, published in *El País* on 11 August 2006, the famous chef Ferran Adrià said:

"One thing that has not been resolved in this country is the protection of creativity. You can copy without fear. R&D makes little sense. The same thing happens in restaurants."

...

"You invent something and a month later, somebody's copying you! In life, there are things that are wrong, things that don't work, and this is one of them. You can work on something for years with hope and ambition and, a month later, someone comes along and introduces it without having put in any effort whatsoever..."

### Recommended reading

*El País*. You can read the whole discussion in the article published in *El País* on 11 August 2006.

Is it really that easy to copy his ideas? Does this mean that his business model cannot work? We can be sure of one thing: his business is working. So what stops Ferran Adrià from running out of customers?

**1)**Firstly, what Ferran Adrià really sells to his customers is not an idea (a recipe) but rather the cooked dish. For the idea to be consumed by his customers, it must be incorporated into a specific cooked dish, just as one does not buy a concept of a car, but rather a specific car.

**2)** Secondly, in relation to the fact that we consume or use products and services that are the materialisation of an idea, it is not enough to have access to the idea, i.e. the "recipe". To turn it into the cooked dish, we must have the skill and knowledge and the right tools. With regards the latter (tools), Adrià himself often says that the public should not expect to repeat the dishes he cooks in his restaurant because home kitchens do not have the right tools. He recommends cooking simple things at home.

Therefore, the investment in the tools that will enable us to replicate the idea puts limits on the possible number of imitators, and hence, on the number of true copies, that is, dishes cooked by professionals to rival his own. This is a fundamental point to bear in mind with any industry. Copying an idea is not as obvious as it seems, i.e. transforming it into a product or service requires some knowledge (be it the expertise that comes with experience or the knowledge gained by study, or both) and investments in machinery, tools, raw materials, etc., which limit the true rivalry in the industry.

> **The professional technician**
>
> This is something that probably occurs in every professional activity. We may be able to change or regulate the taps in our homes, but we will probably not have the tools that a plumber has (buying them just to change a tap every number of years would be excessive), even if we really believe that we have the technical skills to do it.

**3)**Thirdly, as Maskin notes in the case of software – and as is also the case of textile design and software development – culinary innovation is sequential and cumulative: each new recipe is not started from scratch; it is based on previous results. This is something that Adrià himself explains in a series of articles written in conjunction with Xavier Moret and published in August 2002 in *El País*, chronicling his travels to different countries:

> "Trips are now adopted as a method of creation; that is, we go to be inspired, to seek out the sparks that will give us ideas, or specific ideas from other cuisines that can evolve our own cuisine.[...] I think that this approach of knowing what others do is vital in any activity in which you want to evolve."

Thus, innovation does not appear to come from scratch. On the contrary, each time he comes up with a new recipe, it is inspired to a greater or lesser extent by that of his predecessors, whether in the established cuisine of the culinary tradition of his own country or in the cuisine of other countries. His reputation as an inventor of recipes and good executor of them (his reputation, built on the experience of those who have dined at his restaurant) allows him to enjoy what we call in section 1.3. "competitive advantage through differentiation", which means that he can charge a higher price than other chefs (perhaps his imitators) without losing his clientele.

Alternatively, a company can base its competitive advantage on its lower costs, as we saw above with Zara: while perhaps not the most innovative company of its sector, it is inspired by or adapts the designs of other companies with a certain style (i.e., people like to wear the clothing it sells in its stores) and it is capable of doing so at lower costs than its rivals.

# Complementarities

When dealing with software, we need to remember that what we actually value is not the product by itself, but a series of products that complement one another; in fact, the software is simply one of the parts of the system that we actually use.

It is common to see complementarities in products and services related to ICTs.

### Example 1.10. Complementary products

There are televisions with very diverse levels of quality, but even the best television is a completely useless appliance if we have no connection to television channels, DVD player, etc.

**The complementarity of computer equipment**

Similarly, we do not simply want a computer (taking "computer" to mean the physical object, as we saw above with the television), we also want the physical objects that complement the computer, such as printers, digital cameras, scanners, etc. And all these physical objects are not enough; we also need software. We need to have everything that will make the computer run (i.e. the operating system), along with the software we call applications, which allow us to use the computer to perform different tasks. Examples of application software include office automation packages, Internet browsers, e-mail, etc.

Therefore, the complementarity of the various products that make up a system in any digital technology (not only the computer) means that each element of the system in isolation does not really serve much purpose. Naturally, this means that it is essential for these different parts to fit each other and to work properly as a whole, i.e. the various components need to be compatible with one another.

# Network effects

We say that there are network effects or externalities when the value of a product or system for each person who uses it increases the more people who use it. Network externalities can be of two different types:

1) Direct.

2) Indirect or virtual.

## Important

**Direct externality** is perhaps easier to understand: we often find a product more valuable the more widespread it is, since we can then share its use with more people.

### Example 1.11. Direct externality

Obvious examples of this are telephones, fax machines, e-mails, etc. Note that to truly take advantage of the mass of people who also have a phone, it is essential that theirs and ours are compatible (they understand each other). It is pointless for us to have a fax and for others to have one too if their fax does not accept or understand the messages sent to them by our machine.

As we will explain in more detail in the next section, potential network effects are not used to advantage unless there is a standardisation process ensuring that the objects in the hands of different people are compatible, since only then can we really communicate with lots of people.

**Mobile telephony**

In the United States, the various mobile telephony companies could not agree on using the same system. As a result, mobile telephony in the US is much less useful than in Europe, where the European Commission promoted the use of a single standard for all countries. The immediate consequence is that mobile telephony is much less widespread in the United States, to the detriment of the entire industry, companies and clients.

## Important

**Indirect network externalities** are a more subtle economic effect. When a product is actually a system made up of different parts that complement one another and are not very valuable

individually, the value of a product depends on its popularity, since we will have more complements (or better quality parts) the more people who become interested in the product.

In any case, direct and indirect effects have one thing in common: again, it is essential for other individuals and companies to have **compatible products**.

In these cases, to ensure that the markets for these products take off, one of two situations must occur: either the government must intervene or the initiative must be taken by an economic agent with sufficient power to modify the market conditions by itself and sufficient financial resources to withstand years of customer adaptation.

## Example 1.12. Direct and indirect effects

Here are two examples of the importance of these effects for the launch of products with network externalities:

1) The new high-definition video formats. The manufacturers of the new design have secured the commitment of the major film producers, who have said that they will broadcast their new productions in this format. Thus, the customers who use the complement for the new video players will be guaranteed support to make the most of the superior resolution of these appliances.

2) The next example shows that this economic effect is present in other sectors too, not just in ICTs. We will not buy a car that runs on the new biodiesel fuels (i.e. produced from vegetable oils) if we cannot find service stations supplying these; in turn, individual service stations will have little interest in changing their pumps and deposits if they feel that they will not have any customers, manufacturers will not be encouraged to make biodiesel, etc.

In these cases, in contrast to what happens when other people also have fax machines, there is no direct service to be gained from other people having cars that run on biodiesel (i.e. there is no direct effect). Only when there is a considerable mass of people with biodiesel cars will service stations adapt their fuel deposits and pumps to the new fuel. We could say that, indirectly, any person who buys a biodiesel car is doing a favour to other biodiesel car buyers.

Indirect externalities thus explain the importance of the use of this new fuel for growth by the fact that the government subsidises the cost of its manufacture and the importance of the recent agreement between **Acciona**, currently the most technically advanced company in Spain in the manufacture of biodiesel, and **Repsol**, with the largest fuel distribution network in Spain. The agreement between the two companies will ensure the supply of biodiesel fuel at service stations in the near future. Manufacturers and dealers will now be encouraged to sell biodiesel cars because they can guarantee buyers a no-nonsense fuel supply.

When the important features of products and services include complementarities and network effects, the most important consequence of this is that a product will not be viable if we do not achieve a sufficient critical mass of users: below a certain number of users, the product will not offer enough benefits to make it valuable, so the potential suppliers of complementary products will not make the necessary investments to make them available to customers.

**VHS format**

Inertia towards the use of a version can eliminate the viability of alternative versions that are technically feasible. **Betamax** video recorders disappeared when everybody decided to have **VHS** video recorders instead. Even though the total number of households with video recorders increased each year and the number of films available on video also grew, the owners of Betamax video recorders did not have access to them because most new titles only came out in VHS format, which was much more popular. After a time, manufacturers only made the effort to improve the VHS versions of video recorders.

Another danger created by these effects is that a consolidated company with a considerable customer base may interrupt the normal operation of competition through strategic actions that make it difficult or impossible for the new products and services of its rivals to obtain a sufficient critical mass.

In software, as we will see shortly, the main anti-competitive strategy is to make the product of the company dominating the market incompatible with the products of its rivals.

# Compatible products and standards

## Important

We can define a standard as the set of technical specifications allowing compatibility between the different parts of a system.

As we saw in the preceding sections, the value of a product depends largely on the existence of accepted standards:

1. When a product is made up of different elements that complement each other.

2. When the network effects are significant.

In the ICT industry, it is clear that the standardisation of hardware (i.e. the physical devices) has, fortunately, advanced a great deal. Today, virtually any computer peripheral can be connected to a port on a computer (such as a USB port), and when we buy a printer, for example, we know that we need not worry: when we get home, we will be able to connect it to the computer without a problem.

**Component obsolescence**

Those of us of a certain age will remember that things were quite different some years back. We have all had the experience of purchasing an electronic or computer device or part that has become obsolete simply because we can no longer connect it to the other components that it is supposed to form part of.

And the younger ones among us will understand what we mean if they think about all the chargers we have to lug around (mobile, laptop, etc.) because these devices do not work with the same charger – often even when the products are manufactured by the same manufacturer! If we decide to change our mobile one day, we can unfortunately be sure that we will have to throw away the charger because it will be of no use anymore.

# Switching costs and captive customers

Very often, we have products designed to offer a similar service that are unfortunately not compatible with one another. This was the case of records and CDs, and more recently, with devices to play video in VHS and DVD format.

Objectively, in these two examples, we can say that one of the technologies is clearly superior to the other. So if we have to choose between the two technologies with no prior conditioning factors, we would be in no doubt about which to use.

Due to complementarities, however, for those who used the outdated technology, the switch was very expensive at the time. Those with vinyl records who wanted to change to CDs had to first buy a CD player and then buy their records again on CD if they wanted to play them using the new technology.

In general, due to complementarities and network effects in the world of ICTs, switching from one version of a product to a different and incompatible one is expensive, to the point that we will possibly continue to use the old technology for a long time unless we consider the improvement in quality to be very significant.

Naturally, with computers and particularly with software, these switching costs can be significant. They include the costs of learning new programs when we are already used to a given version. This is the reason why programmers tend to make new programs that are similar in appearance and operation to the programs we are already familiar with.

### Example 1.13. Similar programs

The OpenOffice word processor mimics Microsoft Word, which, in turn, imitated an earlier program, WordPerfect, which did the same with WordStar (i.e. the most popular word processor of the time in each case); Microsoft Excel mimics Lotus 1-2-3, which, in turn, imitated a previous program, VisiCalc. And we could continue with many other examples.

Given the costs of switching from one product to another, if incompatibilities arise, consolidated companies with a solid customer base can be tempted to inflate these switching costs, making it harder for customers to switch products or suppliers.

### Example 1.14. Switching costs

Years back, when rival companies emerged, the old telephone monopolies tried to force their customers to change their telephone number if they wanted to switch suppliers (the idea was that customers would not want to incur the cost involved in communicating the change of number to everybody they knew).

Similarly, with software, consolidated companies are tempted to make their products incompatible with those of their rivals.

## Compatibility and standardisation policies within and between platforms

As we have seen, compatibility between the different parts that make up a product and between different products is essential if we are to make them much more functional. Hence, it is important to establish standards that will allow us to make **products compatible** with one another.

Very often, **standardisation** comes about when the format of an essential part of a system is adopted by everybody. This essential part that marks the standardisation process is sometimes called a platform.

These standardisation processes are sometimes the result of the work of bodies set up for the purpose of defining these standards. They can be state or supra-state bodies, or created by members of the industry.

In software, different standards are established for any given procedure, such as all communication protocols governing the transfer of information on the Internet.

Other times, however, a company from the industry controls a portion of it.

### Example 1.15. Sony and Phillips

These two companies were able to impose their technology for producing compact discs through the force of circumstance. As a result, all record labels now distribute their music on this digital format, all music devices are designed to play them, etc.

In software, obviously, the prime example of a platform in the sense we have explained is the Microsoft Windows operating system, installed on the vast majority of computers, both personal and servers.

It is important to understand the interests that guide the owner of a product that has been transformed into a platform in one way or another. In particular, we will look at the interests behind the policies of compatibility between its product and products that complement it (policies of compatibility within a platform) and with products that are its potential rivals (policies of compatibility between platforms).

## Policies of compatibility and standardisation within a platform

Within a platform, a broader range of applications can make the platform more valuable in two ways: customers get more out of the platform – and are thus willing to pay more – and the application creators in turn will see more business opportunities (as there will be a larger potential customer base). As a result, they will make applications to run on this platform, which will attract more clients, etc., creating a virtuous circle that will encourage the dissemination of this product.

Thus, more applications complement the platform and make it more valuable. In theory, the platform sponsor should be interested in opening it up to application developers – indeed, Microsoft often argues that it has an open policy because it shows the parts of the Windows software code (APIs) that application developers need to know for their products to work with Windows.

However, the founder will have conflicting interests:

1) If it also has applications offering good performance, it will want to weaken the performance of competing products and – in the worst case scenario – even make them incompatible with its platform.

2) It may also be concerned that some applications may subsequently become new platforms around which the other applications will develop without depending on the platform that it controls.

> **Microsoft and Java**
>
> This is what happened to the Netscape browser and Java programming language: Microsoft carried out anti-competitive policies against this software because of concerns that it could develop and replace Windows as the software platform for PCs.

To some extent, this gives us an indication of the behaviour that we could expect of the owner of a platform established as the de facto standard when faced with other products that could steal away its privileged position, as we will now discuss.

## Policies of compatibility and standardisation between platforms

We have seen above that, due to switching costs, the share of customers accessible by the company that controls the platform can be a barrier to entry for rivals, when there are network effects, if the company does not make its product compatible with those of its rivals. Naturally, it is not only the rival companies that lose out with these anti-competitive tactics but society as a whole, since the options from which to choose are instantly reduced, and ultimately, so too is the quality of products available, because fewer companies are prepared to spend resources on innovation and product improvement.

### Example 1.16. Incompatible products, anti-competitive tactics

The best-known example of this kind of behaviour is that of Microsoft with its two flagship products: the **Microsoft Windows operating system** and the **Microsoft Office office automation package.** Microsoft clearly does all it can to avoid compatibility with other platforms (particularly with the GNU/Linux operating system, for example). In the same vein, Microsoft has systematically pursued a policy of non-compliance with various standards established by the computer industry by developing its own version of the standard and failing to document the changes it introduces adequately. Very often, when programs and applications apparently do not work properly, it is because the platform does not meet the standards adopted by the industry.

The conflict between the various authorities representing the interests of society (both in the United States and the European Union) and Microsoft basically concerns this purposeful manipulation of the process of standardising a technology, altering the capacity for communication and interoperation between different information platforms.

## Public software policies

We will now briefly discuss some public policies that can promote the proper functioning of software markets, particularly those that allow free software to compete with proprietary software on equal terms and as a valid and viable alternative in cases where the proprietary software boasts the advantage of already having an established mass of users.

**Defence of competition**

First and foremost, governments must guarantee fair **competition** in the software market.

The chief action of the competition authorities should be to ensure that no artificial incompatibilities are created (i.e. ones that do not have a technical explanation) between different technology platforms.

The current conflict between the European Commission and Microsoft boils down to the latter's manipulation of the degree of compatibility between different products by altering the capacity for communication and interoperability across different software platforms, in this case, communication between the operating systems managed by servers and those managed by personal computers.

The European Commission is asking Microsoft to make the information protocols of the Windows operating system available to everybody (particularly computer server manufacturers and programmers) so that the other operating systems can be made compatible with this system, i.e. so that all other operating systems can communicate and interoperate with servers running this operating system.

Naturally, Microsoft's aim is to exploit the fact that the Windows operating system is already widely implemented by artificially raising the costs of switching to another software for its customers.

**Policies for the adoption and support of free software. Enforcing compliance with the standards**

We have seen the importance of network effects in ICTs and the need for software to have a critical mass of users in order to be viable. Through these network effects, large companies can exert their leadership over the implementation of free software. If the government and major corporations (in their own interests or as a service to society) promoted free software in their organisations, they could create a sufficient critical mass for the population to consider the use of free software more accessible.

Much of the proprietary software used today in these organisations could easily be replaced by free software with similar or improved benefits. The only obstacle is the switching cost for individual users because of the lack a sufficient critical mass.

The network effect of this policy in these organisations would be significant, particularly the indirect network effects that would be generated: if these large organisations were to acquire free software, this would create an important source of business for IT companies whose business model is based on free software and the provision of IT services to complement its implementation.

In all events, these organisations must first undergo a process of software acquisition requiring compliance with certain protocols and compatibility standards. If the government, for example, were to establish procedures for the acquisition of software and appropriate computer services, this would probably require the creation of a public agency to advise the various government departments. These agencies could implement different mechanisms to promote the use of free software in government bodies.

# abstract

The information and communication technologies business has specific features that affect the economic model of the business and, hence, the market.

Beyond the creation of value in products and management to gain a competitive advantage over competitors on the market, a company can adopt a particular strategic policy to generate an impact on the economic effects of the market:

- Although production costs are high, the costs of copying are minimal.
- Exploitation of ideas and safeguarding of intellectual property.
- Exploitation of the product's complementarities.
- The net effect of the product, whether by linking its worth to widespread use or as an indirect promoter of complements.
- Compatibility between rival products.
- Control of switching costs in the face of product evolution and customer captivity.
- Introduction of policies on compatibility and standardisation within and across platforms.

Consequently, the particular features of free software allow it to establish a new business format that breaks the mould of the typical policies of a very traditional technology market in terms of the positioning of the competition.

# Bibliography

## bibliography

Boldrin, Michele; Levine, David. 2008. *Against Intellectual Monopoly*. . Cambridge: Cambridge University Press. < http://levine.sscnet.ucla.edu/general/intellectual/againstfinal.htm >

Jaffe, Adam B.; Lerner, Josh . 2004. *Innovation and Its Discontents*. . New Jersey: Princeton University Press

Lerner, Josh; Tirole, Jean. 2002. "Some Simple Economics of Open Source".*The Journal of Industrial Economics*. (pg. 197-234).

Perens, Bruce. (2005, October). "The emerging economic paradigm of open source".*First Monday*.. Special Issue #2: Open Source. < http://firstmonday.org/issues/special10_10/perens/index.html >

Shapiro, Carl; Varian, Hal. 1999. *Information Rules: A Strategic Guide to the Network Economy*. . Boston: Harvard Business Press

**Press**

Adrià, Ferran. (1 August 2002). "Cazadores de ideas".*El País*. .

*El País*. "Aquí unos amigos" (interview with Ferran Adrià, 19 July 2008)..

*The Economist*. "Does IT Matter?" (1 April 2004). .

*El País*. "Es difícil prevenir una burbuja" (interview with Eric Maskin, 29 June 2008). .

<http://people.ischool.berkeley.edu/~hal/people/hal/NYTimes/2004-10-21.html>

*El País*. "El tomo ha muerto, viva la red" (22 July 2007). . Negocios.

*El País*. "Prince vuelve a enfurecer a la industria musical" (15 July 2007). .

"Star Turns, Close Enough to Touch". (12 July 2007).*New York Times*. .

Varian, Hal. (21 October 2004). "Patent Protection Gone Awry".*New York Times*. .

# Chapter 2. The software market

Lluís Bru Martínez

GNUFDL
2010-02-01

# preface

This module introduces the main features of the software market in general and how the free software model adapts to this market.

In the first section, we will see that it is fairly common to have access to products that are freely distributed or free of charge in our environment, and we will look at the particular way in which this business works.

The second section looks briefly at the target market of the software and the most common means through which potential customers acquire the product.

# objectives

After completing this module, students should have achieved the following aims:

1. To understand the features of the market of products with free access.
2. To understand the relationship between free software and the exploitation of parallel business models.
3. To understand the implications of software supply on the business concept.
4. To obtain a detailed knowledge of paradigms of software development and relate them to the features of free software.

# Businesses with similar features to free software

Now that we have looked at the main economic concepts, we can answer the question we asked in section 1.4 "Summary" of module 1:

If free software is free, i.e. by definition, anybody can gain access to this software – possibly at no cost – how is it possible for computer scientists (and computer companies) to earn a living from programming free software? Can we trust that resources (money and people's time) will be spent in the future on its maintenance and development?

## Is it really so shocking that software can be free?

To put it another way, is it really so rare for a product to be freely distributed or even free of charge? If we look closely, we can identify certain business models that are based on offering a product free of charge to customers.

### Example 2.1. Free products

In Spain, we have television channels such as Antena 3, Cuatro, Telecinco and La Sexta that offer free television to viewers. Of course, the business of these stations is to sell advertising, that is, to act as intermediaries between companies that want to publicise their product and their potential customers (for example, viewers will see advertising placed before, during and after the broadcasting of a football match).

In general, any company whose business is to act as an intermediary between other companies and their customers must decide what pricing policy to adopt, and perhaps the best option is to dismiss the possibility of making money with some of these customers.

### Example 2.2. Different business models based on free supply

If a television wants to earn revenue from advertising, it needs to guarantee its paying customers (the companies that place advertisements during broadcasts) the largest possible number of viewers, and the best way to do this is to allow the latter to receive the television signal for free.

Similarly, if **Adobe** wants to attract customers for its PDF file creation product, Adobe Acrobat Professional, it makes sense to offer the simplified version of this software, Adobe Acrobat Reader, for free. This way, Adobe can guarantee its paying customers that other users can actually read the documents that they create.

Likewise, **Amazon**, besides being a book shop that sells on-line, has transformed its website into a platform that connects its customers with second-hand book shops offering used books at a discount. When we check the availability of a title, we see Amazon's offer together with that of the other bookshops. In this case, Amazon offers its customers the possibility of viewing the series of available books for free and instead charges the bookshops for its intermediation service. Given the reasons for adopting this pricing policy discussed earlier, it is convenient that Amazon earns money from the sales of the other book shops because it might otherwise be tempted to offer a biased service (ensuring the sale of its own books over those of its rivals listed on the website).

Alternatively, a company can offer customers a product for free, but link it to another product, which is the one it wants to sell. An example of this follows:

### Recommended reading

*El País*. You can read the full article published in *El País*, 15 July 2007 "Prince vuelve a enfurecer a la industria musical".

> "Anyone who has purchased the British weekly *Mail on Sunday* this morning has taken home a free copy of Prince's new work, *Planet Earth*. In all, 2.9 million copies have been sold."
>
> [...]
>
> *Planet Earth* will also be distributed free of charge to those attending any of the 21 concerts that the Minneapolis musician is putting on at London's O2 Arena from 1 August to 21 September."
>
> *El País*, 15 July 2007.

As we can see, in the first case, it is quite possibly the newspaper that has bought the right to give away copies with its publication (a way to promote the newspaper), while in the second case, the artist foregoes the possibility of making money with the distribution of copies of the CD (contrary to the efforts of music labels and record shops who want to hold on to their business model at all costs) to focus on making money from his concerts. (Another story, this time in the *New York Times*, says that the musician is putting on

exclusive concerts at small venues for which tickets, with meal included, are being sold for $3,000 (12 July 2007, "Star Turns, Close Enough to Touch").

### Recommended reading

*El País*. You can read the full article published in *El País* on 12 July 2007 "Star Turns, Close Enough to Touch".

# Software as part of a product

Software is only one component of a product (albeit a very important part), a part or complement of the whole product that we wish to obtain, and what we want is to have all the pieces – such as the computer and the software – at the same time.

As a result, the multinational giants of the computer industry like IBM and Sun Microsystems provide funding to computer scientists who develop free software. Their selfish (in the sense that they are thinking primarily of increasing their profits) reason is that they think that this will increase the sales of complementary products and services for which they charge their customers.

Likewise, the leading mobile phone manufacturers (Nokia, Motorola, Siemens, Samsung, etc.) teamed up to create – and allocate financial resources to – the Symbian consortium, which develops free software designed as a program to operate the mobile telephones that they manufacture. Thus, all mobile telephone manufacturers use the same platform (the same operating system), which is based on the **GNU/Linux operating system** and is flexible enough for each manufacturer to then design a different mobile phone model to its rivals, incorporating improvements and variations to attract customers (telephones that double as cameras, allow the user to send e-mails, etc). Each company changes the appearance of the phone screen to adapt it to the services it offers, since it has access to the source code of the program used to operate the telephone. This system encourages innovation and product improvement because the companies expect to attract new customers by creating a device (the telephone) that works better than that of its rivals.

The fact that the big multinationals have fully incorporated free software as a tool in their activities thus guarantees the future development of this software. It even ensures that IT engineers can, on their own initiative, engage in the development of free software. As Lerner and Tirole (2002) explain, these engineers can demonstrate their professional expertise to companies in this sector by participating in the improvement of this software, which will make them highly sought after by IT companies, hence allowing them to improve their employment prospects.

# Software supply. Distribution

Just because the software is free, this does not mean that we cannot have companies that exclusively supply related IT products and services.

To begin with, one possible business is the distribution of free software. In addition to selling CDs containing the free software, these companies provide technical support to the consumers and businesses that opt to use free software (Red Hat is the best-known example of a company that has developed this line of business). Therefore, the company offers its experience and knowledge of the software to clients, guaranteeing them any technical support they may need.

If we think about it, this business model is perhaps not as uncommon as it might appear. For example, the publishing house Aranzadi has created a very similar business model.

## Aranzadi

Aranzadi offers its clients (legal professionals) a comprehensive source of legal information. It also provides the technical support needed to process all of this information efficiently.

The information has always been freely available (Spanish legislation is published in the Official Gazette and all law firms subscribe to it). However, organising the information in useful ways is a complicated task, and this is the service that these publishing houses offer to their clients. And, naturally, these companies have incorporated digital technologies to serve their clients, as we see in the following press release:

> The offices of law firms and tax experts are still bedecked with yard upon yard of solemn legal tomes. But these are increasingly becoming mere decorations. Most legal experts are already opting to access the necessary documentation for their work through the Internet, an out-and-out revolution sparked by the big legal publishing houses such as Corporación El Derecho, which has set a benchmark in new technologies.
>
> Corporación El Derecho provides legal information to state prosecutors (through a call for tenders organised by the Spanish Ministry of Justice) and basic tax information to the Tax Office.
>
> *El País*, 22 July 2007.

### Recommended reading

*El País*. You can read the full article published in *El País*, 22 July 2007 "El tomo ha muerto, viva la red".

# Software supply. Service

Broadly speaking, an IT engineer who works with free software has a similar profession to a chef, car mechanic, plumber or lawyer.

Law firms work with a knowledge and understanding of legislation that is as free and widely available as free software could be. Clearly, their business model consists of raising revenue from a complementary product, which is their expertise or in-depth knowledge of the law, their ability to adequately organise the information set down in legislation to defend their client's interests, which are things that their clients cannot necessarily do.

Ultimately, the lawyer incorporates the right ideas into the right product for its client (defence of the latter's interests).

Similarly, computer engineers who work with free software offer clients their expertise, the ability to meet their need to organise information in a specific way and process data by harnessing the intrinsic possibilities of the free software available, or, if necessary, by developing additional code.

Thus, we can see how a given economic sector (legal services) can even have different levels of information (corporate, law and Aranzadi on one level and law firms on another), which gives rise to multiple business models that simultaneously coexist.

# Who needs software?

## Software, a basic need in any company

Who are the clients of software companies? Nowadays, potentially any company. As Nicolas Carr points out in "IT doesn't matter", ICTs have been incorporated as an essential tool for all companies, just as nowadays all companies are connected to the mains to light up their offices and power their machines, they are all equipped with telephones, or they all use cars and trucks on the motorways to transport their raw materials and products.

### Additional reading

N. Carr. (1 April 2004). "Does IT matter?".*The Economist*. . < http://www.nicholasgcarr.com/articles/matter.html >

When Carr writes in his article that "ICTs no longer count", what he means is that a company no longer has a competitive advantage just because it uses them, since all companies now have access to them.

### Example 2.3. On-line ticket bookings

A commonly cited case in this regard are the commercial airlines that developed the first ticket booking software. At the time, this software gave them an important edge over their rivals. Today, all commercial aviation companies have a website where we can make bookings and purchase plane tickets, so this software no longer constitutes an advantage for a company over any other.

This evolution in the use of ICTs can be an advantage for free software development in that it reduces the possibility for companies to get carried away with the idea that having proprietary software for their internal processes can give them a competitive edge. Given that any company can obtain software with similar capabilities, it is probably best to use free software that can incorporate the developments made in other activities and tailor them to the specific needs of the company.

# Paradigms of software development

We said in the previous section that all of today's businesses need to use ICTs and software in particular, but how can a company get the software it needs for its production processes?

Based on the classification developed by Bruce Perens in "The emerging economic paradigm of open source", we can sort companies as follows:

### Required reading

B. Perens. 2005. *The emerging economic paradigm of Open Source*. . < http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1470/1385 >

**1)** The Microsoft and Adobe model (Perens' "Retail" model), whereby a company develops software and sells it packaged to customers.

Thus, from their point of view, customers can forget about the development of the software and simply buy it finished.

## Consequences of the retail model

Naturally, this software development usually takes the form of proprietary software (where the provider does not reveal the code to its customers). From the point of view of somebody who purchases this software, the first obvious drawback is that it is not designed for his/her specific needs (because, obviously, it has to be sold in a very uniform way in order to be of interest to a range of customers). Another potentially serious problem is, as we mentioned earlier, the danger of being trapped by the provider, which makes it difficult to switch to other software, retrieve certain databases, etc. Conversely, but with similar consequences, there is the danger that the provider will disappear and thus cease to provide the required software maintenance and improvement services.

**2)** The business model where the company that needs the software develops it, either with the computer scientists on its staff or by hiring a specialist IT company to develop it (Perens' "In-House and Contract"model).

### Development cost

Of course, this way of developing the software that a company needs can be very expensive, and can lead to repeating parts of programming that have already been developed and could have been used.

In the last two models of development in Perens' classification, companies seek out other companies with which they can collaborate to develop the software they need.

**3)** In this model, the consortium develops a software that is not free (i.e. that will not be available to companies that do not participate in its development).

**4)** In the last model, the consortium companies develop free software, i.e. with a source code available to any other company, even if they are not involved in its development.

This offers the clear benefit of being able to take advantage of improvements in the community of programmers created around the project, thus reducing development costs.

Of course, the development of the free software will not be free to the consortium companies, which will need to finance an initial group of programmers. The danger of consortiums (both proprietary and for free software) is that there is a lack of leadership in the development of the project because no company wants to commit to guaranteeing its development, which creates a barrier to its implementation (either from the start or when successive developments generate new expenses).

# abstract

In our more immediate environment, a scenario is being shaped by multiple business models converging with different policies to achieve their aims, from the direct promotion of the product per se to the supply of products free of charge to encourage customers to access a new world of complementary products and services.

The free software business uses the latter market form, setting up parallel and complementary businesses based on its promotion. Nowadays, many companies and multinationals have adopted a clear stance in support of the development of free software, especially considering that software is a basic product for any business and that the free software development model offers guarantees for securing these aims.

# Bibliography

# bibliography

**Karminski, D.** (1999). "Core Competencies: Why Open Source Is The Optimum Economic Paradigm for Software". <http://www.doxpara.com/read.php/core.html> [Consulted in February 2009]

Perens, B.. 2005. "The Emerging Economic Paradigm of Open Source". Published in*First Monday, Special Issue # 2, 3/10/2005*. . Cambridge: Cambridge University Press. < http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1470/1385 > [Consulted in February 2009]

**Press**

*El País*. "Prince vuelve a enfurecer a la industria musical" (15 July 2007). .

*New York Times*. "Star Turns, Close Enough to Touch" (12 July 2007). .

# Chapter 3. Software as a business

Irene Fernández Monsalve

GNUFDL
2010-02-01

# preface

In this module, we will look at the "classical" view of software as a business. We will focus on proprietary software, leaving the study of the further possibilities of free software in this scenario for a later module. Although some of the aspects that we touch upon will be irrelevant when it comes to the application of free software strategies, others will still be valid to a large extent.

We will review some of the key factors to consider when designing a business around software, such as the **choice of the main activity** and the general approach of the company (selling products or services), aspects of **selling** and **marketing** (how to choose our market and how to approach it), and the **definition of its products or services** (what type of products or services to develop and how to position them).

# objectives

After completing this module, students should have achieved the following aims:

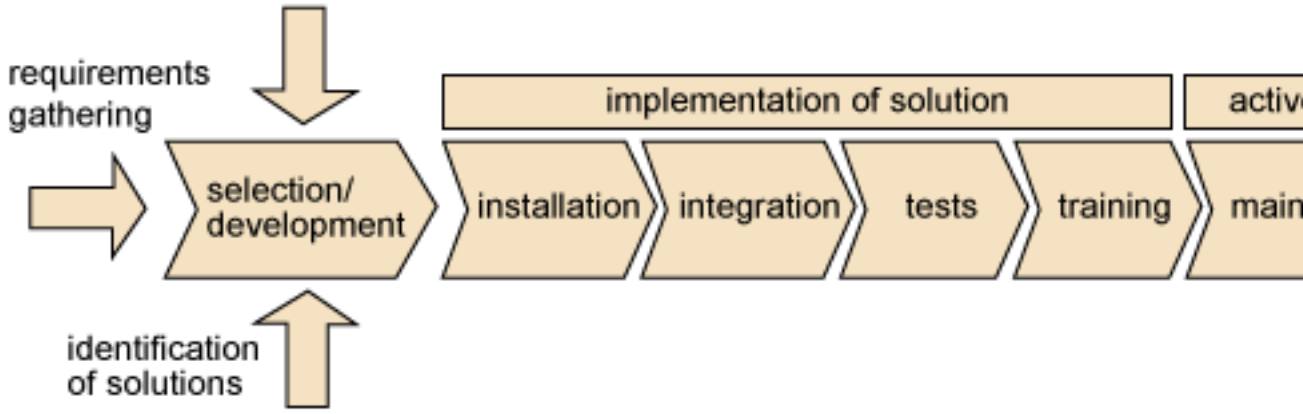
1. To obtain a global vision of the business opportunities of software.
2. To learn about the traditional models of software companies.
3. To understand the economic features of and differences between product companies and service companies.
4. To identify the key factors that software companies need to consider when positioning their products on the market.

# Business opportunities with software

Both individuals and corporate environments have software needs that generate multiple business opportunities.

The basic task involved in meeting these needs is to create this software, the task of development per se. However, the needs to be met do not end here; this is only the beginning. Once the product is made available, a number of related needs arise, such as consulting, installation, configuration, maintenance, support and training, for which certain customers (mainly other companies) are willing to pay.

Throughout the process of technology adoption, from the identification of needs to the decision to build or buy, right up to the end of the useful life of the technology, multiple needs are generated that can be met by many different companies:




Moreover, the process of software creation itself can be interpreted in two ways: as the creation of a product or as the provision of a service. The choice between the two will be critical for defining the company's operation and its potential generation of revenue, which will result in very different business models.

This choice – developing software as a product or a service – also reflects the first issue that a company that consumes software will need to evaluate when adopting a technological solution: whether to purchase a standard, packaged product or to obtain a tailored development.

We can therefore distinguish between the following business activities in relation to software:

- Application development
  - As a product: standard solutions (shrink-wrapped)
  - As a service: custom development
- Provision of services around one or more applications
  - Consulting
  - Selection
  - Installation
  - Integration
  - Training
  - Maintenance and support
  - etc.
- Software as a service

This classification is intended to be neither exhaustive nor exclusive, that is, many companies will implement hybrid models allowing them to provide integral solutions to their customers.

The features of software companies and their business dynamics will vary greatly depending on the activities that they focus on, as we shall see later, but any of the models has the potential to generate both viable and highly profitable businesses.

# Service companies

As we explained earlier, companies can specialise in one or more aspects of the chain of technology adoption and implement a number of activities at the same time.

Hence, for companies that include various services in their business offer, we can distinguish between two types of specialisation: vertical and horizontal.

## Vertical specialisation

### Important

Broadly speaking, companies whose main activity is development will tend to have a **vertical specialisation**. If there business strategy is centred on custom development, their activities will naturally include other related services, such as **installation**, **integration** and **training**. However, as we shall see, companies that adopt the strategy of software as a product will also do well to exploit associated services as a way to guarantee a steady flow of income.

| | **Package 1** | **Package 2** | **Package 3** | **etc.** |
|---|---|---|---|---|
| Development | X | X | | |
| Installation | X | X | | |
| Integration | X | X | | |
| Certification | X | X | | |
| Training | X | X | | |
| Maintenance and support | X | X | | |
| Migration | X | X | | |

Interestingly, a company that invests a certain amount of money in software licensing expects to invest additional sums in related services, such as maintenance and support, and in updates. Thus, selling products to business clients will open the door to obtaining service contracts with the same clients and hence, a more consistent flow of income over time.

## Horizontal specialisation

### Important

In contrast, companies that exploit the needs generated by the general use of software products will often offer services in a variety of packages, focusing on one or more of the phases of the adoption of a technology.

| | **Package 1** | **Package 2** | **Package 3** | **etc.** |
|---|---|---|---|---|
| Selection/Custom developments | | | | |
| Installation | | | | |
| Integration | | | | |
| Certification | X | X | X | X |
| Training | X | X | X | X |
| Maintenance and support | | | | |

| | Package 1 | Package 2 | Package 3 | etc. |
|---|---|---|---|---|
| Migration | | | | |

Although some companies specialise in training or support, service companies often touch on a number of the phases described, generating typologies such as consulting (with an emphasis on selection, advice, and/or certification), or integral solution providers, which cover all categories, including custom developments and even the provision of hardware.

Companies that create GNU/Linux distributions use a **service provision model with horizontal specialisation**.

### Example 3.1. Example of horizontal specialisation

**Canonical**, creators of the distribution based on **Debian Ubuntu**, perform a task of selection and horizontal integration that encompasses a complete operating system along with several applications, with the basic aim of providing a distribution that is easy to use, install and set up under the slogan "linux for humans". However, since Ubuntu is free software, Canonical's income comes from related services, namely support, training and certification.

These service-oriented companies often observe that their clients prefer to receive integral solutions and deal with a single technology solutions provider. To be able to offer this comprehensive type of service, companies often need a powerful infrastructure and technical capacity, which limits the entry of SMEs as they are unable to meet every single need by themselves.

A common solution is for the service company to contract out the parts that it cannot handle alone. Another very interesting solution is the "pyramidal model of consulting" proposed by Daffara (*Sustainability of FLOSS-Based economic models*), which we will now explain.

### Recommended website

For more information about the "pyramidal model of consulting", see: http://www.cospa-project.org/Assets/resources/daffara-OSWC2.pdf)

Generally speaking, computer support and maintenance can be said to follow the 80/20 rule: 80% of queries are easy and can be resolved immediately. The remaining 20%, however, are important problems and account for 80% of the effort. Hence, a service SME could take care of a high number of clients, dealing with 80% of their incidences and earning a reasonable amount for the service. To solve the remaining 20%, it will require the technical services of the software creation companies, who will obviously need to be paid more than what the company receives from each client but less than what it earns from all of these clients together.

This model will generate sustainable cooperation between the development companies, with vertical specialisation, and the companies offering integral solutions. The former will be able to reach more users through the horizontal consulting firms, which will also mean a significant source of income. The latter will be able to manage a large customer base and provide quality support for a range of products, maintaining a profitable business so long as the customer base is big enough.

## Development companies: to create products or to provide services?

As we said earlier, a company that hopes to focus on development will have two main options to choose from: it could generate **standard products**, packaged to sell to the mass market (**shrink-wrapped**, as they are called), or it could generate **custom developments**, tailored to the needs of individual clients.

The first option has the potential of generating large profit margins but they will be difficult to maintain over time, and it has barriers to entry that could prove unsurmountable. The latter is a far more labour-intensive option with much lower profit margins, but it offers more possibilities of generating constant sources of income over time and of being less sensitive to changes in the macroeconomic environment.

We will now describe the differentiating features of these two options in detail.

**Economies of scale and the possibility of large profit margins**

### Required reading

M. Cusumano. 2004. *The Business of Software*. (Chapter 1, "The Business of Software, a Personal View").

The economic process of software creation has special features not seen in other industries, affording it huge **positive returns to scale**.

On the one hand, commercial companies need to invest large sums of money in development before they can create a commercial version of a product to release, and they must often invest again every two or three years to maintain a constant flow of income. Since the aim of this development is to generate a standard product, it is very risky because there is no certainty that the investment will be recovered later through sales. However, once they have a finished product, the marginal cost of each additional copy sold is next to nothing. The first copy of the software created is very expensive, but the rest costs virtually nothing.

This leads to huge economies of scale on the supply side, which combine with significant economies of scale on the demand side: both due to the time invested in acquiring the skills to use an application and the possible incompatibility of formats, switching from one product to another is a difficult and expensive task. As a result, the bigger the user base of a product, the easier it is for this base to grow and survive over time. In the software market, then, we can come across "winner takes it all situations", where huge profits are generated and the entry of new companies to these markets is simultaneously blocked.

### Example 3.2. Examples of companies that generate standard products

The companies that have exploited these large economies of scale include some of the giants of the software industry, such as **Microsoft**, which tops the desktop operating systems market, and **Oracle**, with its purchase of **PeopleSoft** in 2005. However, there are small companies too, known as independent software vendors (ISV), that produce feasible businesses by exploiting specific niches. Examples include Pretty Good Solitaire, developed by the two-staff micro-enterprise **Goodsol Development Inc.** (one of the most popular solitaire games), and HomeSite, a HTML editor developed by the Bradbury Software micro-enterprise in 1995, which was purchased by Allaire Corp. (Allaire was later purchased by Macromedia, which, in turn, was absorbed in 2005 by Adobe).

In contrast, a company that engages in custom development will not have access to the economies of scale of standard software. Every new customer will require a specific development, making it a costly investment in time and effort, although this type of company does tend to reuse its developments where possible.

## Need for initial investment

When we set up a company based on the traditional product idea, we come across an important problem: the **need for initial investment**. During the early stages of the company, dedicated to development, there will be no income flow, but there will be expenses until the first versions of the software are ready for release. Besides the expenses deriving directly from development, we need to take into account the necessary expenses of marketing and sales. There are two solutions to this problem: **obtain external investment**, **or start another type of business activity** that generates sufficient income to allow for simultaneous development of the product.

### Important

**Custom development companies** entail **much less risk** and can start their activity with a much smaller investment (development only begins once a contract is signed), thus avoiding the need to search for outside investors.

The financial literature tends to focus on the discussion of companies that finance their development from venture capital investments since they are more attractive. This type of financing allows for faster growth, which is an important factor in success according to Cusumano. (Michael Cusumano, *The Business of Software*)

### Note

At this point, we can consider the following: what parameters do we use to judge the success of a business initiative? Investors and financial publications consider a successful company to be one that manages to make a profit every year, and probably those that display growth too. A company that remains the same size with an income statement showing no profit will not attract the attention of investors or the financial literature. However, a company of this nature may have been very successful in creating quality jobs and maintaining them over time. For many entrepreneurs, this can be the main aim.

Obtaining sufficient outside investment can be an insurmountable obstacle and, even when it is possible, it has certain disadvantages that we need to take into account. The presence of investors will put pressure on the management decisions of the company, as it will have to generate sufficient profits to repay the investment and make gains. This situation will limit the autonomy and decision-making capacity of its founders.

The other option is not straightforward either. The company would have to redirect its business to services in an attempt to generate sufficient revenue from them to allow for the simultaneous development of the product. As we shall see later, it is difficult to be successful in this through the provision of services because the profit margin is smaller. Moreover, the lack of economies of scale and the presence of competition restrict the possibility of keeping prices high enough.

### Important

In this context, free software emerges with new features to alter the scenario. The possibility of **cutting costs** through the collaboration of volunteers, together with the new schemas of distribution and marketing offered by this collaboration constitute a relevant disruption of these scenarios and have the potential to significantly reduce the initial investment required.

We will look at these aspects in more detail in the following modules.

## Maintaining the revenue stream

One of the basic questions that any company needs to ask is not only how to raise revenue at a given moment, but also how to maintain it over time. While continuity will be the norm for companies that focus on providing services (generally, if clients are satisfied, they will continue to need the services), in companies that focus on the production of standard solutions, the maintenance of a steady stream of income will be fraught by a range of problems.

### 1) Software cycles

Cusumano, in *The Business of Software*, compares the process of writing a successful software product to writing a best-seller. Doing so will generate huge profits but it is also very difficult and only occasionally

generates the latter. The natural life cycle of a commercial software product will eventually cause it to lose the ability to generate income.

Initially, early versions will have several flaws and their functionality will not be finely tuned to the needs of users. This will allow the company that created it to maintain its income over time with the launch of new versions that gradually incorporate improvements into the product, both through debugging and by obtaining much more information on requirements from user and customer feedback.

Predictably, if new versions of the product contain sufficient improvements and are more attractive than the previous ones, they will continue to generate revenue. However, once users decide that the application is good enough, their motivation to pay for a new version will wane. Similarly, trying to maintain the income obtained from a best-seller with sequels has only limited effectiveness.

There are strategies to combat these trends and maintain a stream of revenue through the licensing of successive versions, typically at the expense of consumers. The total or partial incompatibility between successive versions of the product, coupled with intensive campaigns to publicise it, will lead to a new situation of economies of scale on the demand side in favour of the latest version, which will force many users to change even though the previous product met their needs.

However, due to the nature of some software products, constant updating is necessary due to changing user needs.

### Example 3.3. An illustrative example: accounting, labour and tax management applications.

Tax and labour legislation changes often, which means that users need to update their application every time this occurs. As a result, revenue can be kept constant over time because of cyclical adjustments in the financial system and legislative framework.

In addition, once the initial idea has been exploited and studied, it will pave the way for other companies to start producing similar software without having to spend time on R&D or requirements analysis. If they can make the product more quickly, perhaps streamlining it and maintaining only the basic features, they will be able to compete for the same market at a better price. Once enough companies enter this market, generating products that can be interchanged with one another ("commoditisation"), we reach a unique situation: in the absence of other differentiating factors, consumers will buy the cheaper product, which will generate a highly competitive situation.

This phenomenon is common to any type of product and should also be possible with software. However, certain factors protect the dominant companies in this process, which would ideally lead to greater technological diffusion and bring benefits to users (albeit making it more difficult for companies to obtain large profit margins). As explained above, there are some strong economies of scale on the demand side so it will not be as easy for users to consider competing products as true replacements. Moreover, the use of proprietary formats creates an important captive situation that is difficult to escape from.

Hence, free software emerges as a driving force for a situation in which **free goods are perfectly interchangeable**: the appearance of a similar product that is distributed freely or even free of charge makes it more difficult to maintain high revenues from licensing and may be one of the few ways to break the captive inertia generated by proprietary software.

**Free software as disruptive technology**

## Important

The term *disruptive technology*, coined in 1999 by Clayton M. Christensen, refers to innovations that, for their low price and features or due to their focus on a new type of customer, manage to displace the previous market solutions. Free software could thus constitute a disruptive

technology, given the possibility of obtaining it for free and its ability to contribute to the widespread use of software through existing technology gaps.

Though it would considerably limit the possibility of maintaining high profits from licensing, the transformation of the software industry into a scenario of interchangeable goods (commoditisation) could open up new markets, generating an ecosystem of needs around the new interchangeable and widely adopted product.

### 2) Dependence on economic cycles

Traditional software companies with their product focus can generate huge profits but also suffer major losses during unfavourable economic cycles. Despite being consolidated businesses with established products, between 2000 to 2002, many software companies lost 80 to 90% of their value; even Microsoft lost two thirds of its value (Michael Cusumano, *The Business of Software*).

During adverse economic periods, consumption falls and software products are the first to feel the effects. Users simply stop buying software, which can have a serious effect on the product companies that depend entirely on this source of income. Consequently, it is difficult to find a product company solely of this nature, as the guarantee of its income would be too precarious and unpredictable, and would inevitably suffer in harsh times.

Although any business activity will be affected in such scenarios, companies focusing on services are more capable of maintaining their income due to their long-term contracts and clients – who are mainly other companies and, albeit to a lesser extent, will still need to maintain their infrastructure. In many cases, these infrastructures allow the client company to operate more efficiently, thus increasing its chances of survival in difficult times. As a result, it continues to spend on new technology services.

## Hybrid models

In actual fact, there are many hybrid models that combine the sale of standard products and the provision of services to varying degrees in an attempt to reconcile the two trends. We can consider that the degree to which a company leans towards products or services is indicative of its own life cycle, and there is a widespread trend of a transition to services.

### Example 3.4. Example of a hybrid company

Consider a company that starts with a pure product model, obtaining high sales and large profits, but which discovers that it is going to be difficult to maintain this level of income. To ensure its continuity or in response to difficult economic times, it may begin to arrange service contracts with some of its customers, witnessing a significant slowdown in the company's rate of growth but obtaining greater long-term stability. The company may eventually put its entire emphasis on to services, having already saturated the market of its original product.

Of course, this is merely a theoretical example, and many companies will not complete or even begin this cycle at the same point.

Furthermore, the transition to services is not an easy one and can have negative consequences if not done carefully. Adopting a hybrid model in response to a crisis, without carefully considering the business strategy, can cause many problems for a product company.

In times of lack of revenue, the company may cede to pressure from different customers to develop highly specific product adaptations that are difficult to integrate with the main standard product. If this practice becomes widespread and the company intends to maintain its revenue through the sale of the standard product, it may encounter difficulties in maintaining compatibility between the new versions released and the specific adaptations for different customers. The work of debugging and development will multiply and can sometimes generate more expenses than revenue for the business.

# Software as a service

The concept of "software as a service" (SaaS) originated in 1999 as a new way to implement software with an emphasis on functionality.

## Important

The basic approach of this idea is that software is important to users insofar as it allows them to solve a problem, i.e. to the extent that it provides them with a service.

Under this paradigm, the need to acquire a software product, have a related hardware and software infrastructure and the installation and support that this requires would be little more than a hindrance to the end user, who has to put up with them in order to obtain the desired functionality.

Under a software as a service model, all of these problems disappear and the software changes from being a product that can be acquired to becoming a service that can be provided. In this sense, it is important to distinguish between the service companies that we described earlier, which **provide software services** (installation, maintenance, etc.), and this new model, which **provides software as a service** (provision of the specific functionality of this software).

To implement this concept, the provider would take care of all the necessary infrastructure, hosting the required software and offering the service on-line through a browser. A sufficiently powerful communications infrastructure is required but the other technological requirements on the receiver side of the service are reduced, allowing the attention to be focused entirely on the functionality offered.

### Example 3.5. Providing software as a service

More and more companies are using this model to provide enterprise software, such as 37signals with Basecamp (project management tool), and the popular Salesforce.com (CRM or customer relationship management), which allows the software to be tailored to customer needs.

The software as a service model is a low-cost way of providing software to companies, in comparison with the traditional method of selling products. On the one hand, customers save considerable sums on IT infrastructure maintenance and, on the other, providers can offer lower prices because they combine the recurring revenue obtained from the provision of a service and use a single instance of their application at any one time to service a large number of customers.

### Example 3.6. On-line software

We can also find several examples of web applications aimed at private consumers, although this trend is referred to as "Web 2.0". Many have had great success, such as the numerous Google applications and e-bay.

The presence of both free software and SaaS offers is threatening traditional software vendors, who are feeling the pressure with the entry of these new competitors and will have difficulty maintaining the prices of their products.

Software as a service providers also stand to gain a great deal from the use of free software. On the one hand, using it in their software infrastructure will save them significant sums in licensing or development and, on the other, some companies are using free, GPL-licensed applications to develop their critical business applications, keeping their modifications closed as a way to protect their business differentiation. In this case, they are exploiting a loophole in the GPL: modifications of the code must only be redistributed if the program is redistributed. In the case of software as a service, only the functionality – not the code – is redistributed, so the company has no obligation to share its improvements.

# Dominant companies in the sector

As we have seen, orienting a business towards products or services will generate very different business dynamics although both approaches can generate profitable business models. Nonetheless, it will be very difficult to keep pure-product companies alive and the barriers to entry will be substantial.

The "Software 500" survey of "Software Magazine" (www.softwaremag.com), which produces an annual ranking of the top 500 commercial software companies by revenue, shows that both types of company discussed here can be found among the most profitable companies.

However, of the top twenty, only four have a marked product focus with services representing less than 30% of their total business: Microsoft Corporation, Oracle, SAP and Symantec, which offer leading products in their sectors to corporate customers and mass markets (desktop operating systems, databases, ERP and security, respectively).

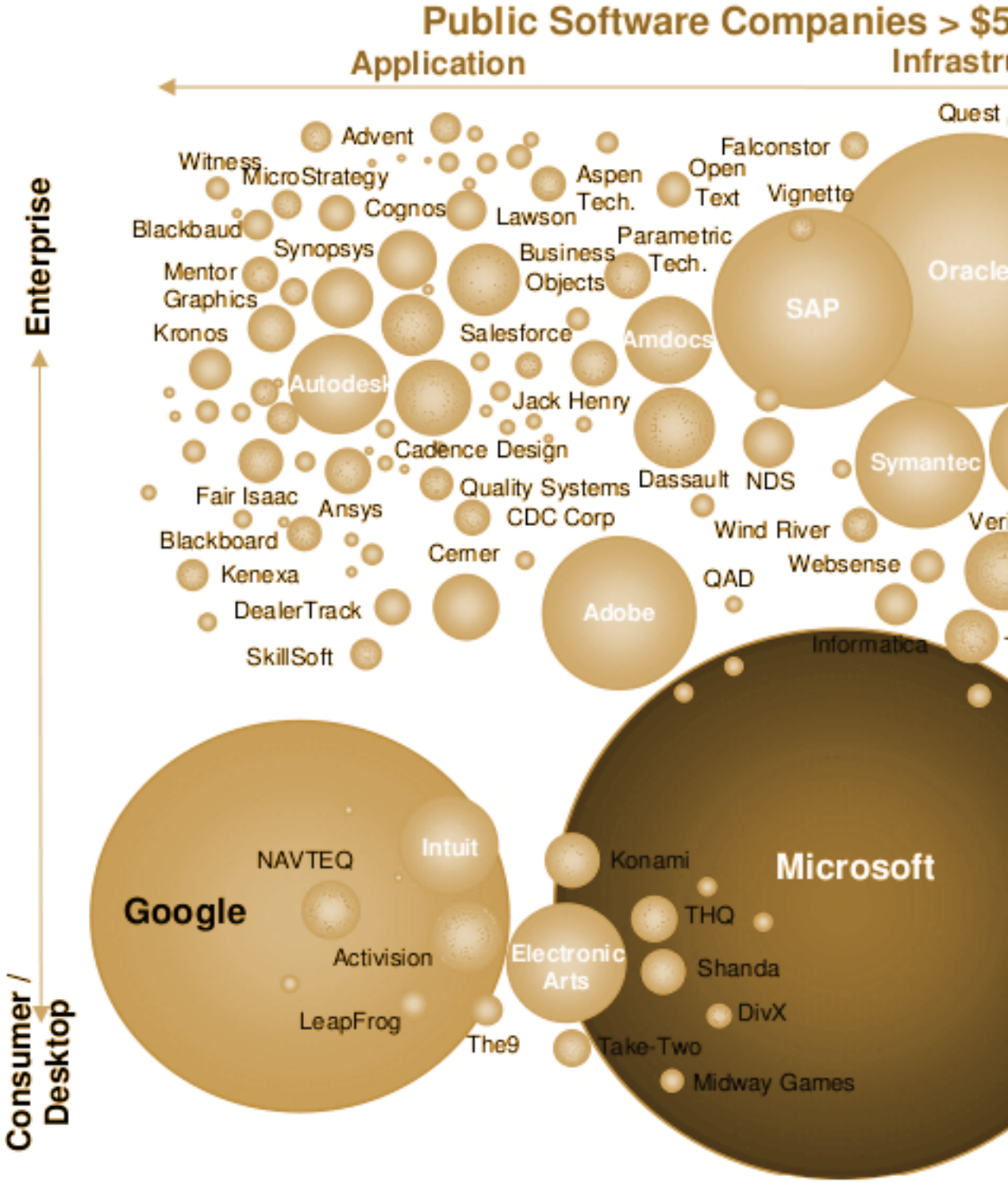
Two companies, Lockheed Martin Corporation and EMC Corporation, have a 50% balance between products and services. Of the remaining companies, ten state that their primary business sector is integration, consulting, and outsourcing services, while the rest, although dedicated to the development of specific products, derive their income primarily from the provision of related services.

## Note

| | Company | Website | Revenue from software /services ($ million) | Revenue growth (%) | Services as a % | No emplo |
|---|---|---|---|---|---|---|
| 1 | IBM | www.ibm.com | $66,451.00 | 3.0% | 72.6% | 394, |
| 2 | Microsoft Corporation | www.microsoft.com | $39,317.00 | 9.0% | NA | 71,( |
| 3 | EDS | www.eds.com | $21,268.00 | 8.0% | 100% | 118, |
| 4 | Hewlett-Packard Company | www.hp.com | $16,918.00 | 2.0% | 92.3% | 156, |
| 5 | Accenture | www.accenture.com | $16,646.40 | 7.0% | 100.% | 140, |
| 6 | Computer Sciences Corporation | www.csc.com | $14,615.60 | 4.0% | NA | 79,( |
| 7 | Oracle Corporation | www.oracle.com | $14,380.00 | 22.0% | 19.7% | 56, |
| 8 | SAP | www.sap.com | $12,309.70 | 23.0% | 29.2% | 39,3 |
| 9 | Cap Gemini | www.capgemini.com | $10,158.60 | 23.0% | NA | 67,8 |
| 10 | Hitachi | www.hitachi.com | $9,019.20 | 5.0% | 85.4% | 356, |
| 11 | Lockheed Martin Corporation | www.lockheedmartin.com | $8,992.00 | 10.0% | 51.2% | 140, |
| 12 | Science Applications International Corporation (SAIC) | www.saic.com | $7,775.00 | 8.0% | NA | 43,6 |
| 13 | NTT Data Corporation | www.nttdata.co.jp | $6,685.80 | 4.0% | 7.9% | 21,3 |
| 14 | EMC Corporation | www.emc.com | $6,014.50 | 16.0% | 51.2% | 31, |
| 15 | Affiliated Computer Services, Inc. | www.acs-inc.com | $5,353.70 | 23.0% | NA | 58,( |
| 16 | LogicaCMG plc | www.logicacmg.com | $5,221.40 | 65.0% | NA | 40,4 |

The following figure illustrates the current positioning of these and other software companies by approach (application, infrastructure, services) and the type of customers they target (business or domestic consumers).

## Note



# Marketing in business: who to sell to?

Thus far, we have looked at several aspects of the basic nature of software companies and the definition of their core activities. However, another fundamental aspect that any company needs to ask itself is what to sell and who to sell to.

## Niche and mass markets

For any company with strong economies of scale, as is true of software product companies, the bigger the user base, the higher the profit margin. Therefore, the seemingly more lucrative option would be to aim its products at mass markets.

However, a strategy like this can be fraught with difficulties: the mass market will be more closely analysed, controlled and saturated by the big corporations. For a company that is just starting out, it will be extremely difficult to compete with companies that are already established and dominant in the sector, and which will also have a large capacity for marketing and diffusion.

It will be easier to meet the needs detected in **niche markets**, which are unattractive to large companies due to their size. For large companies, the potential returns from these markets are too low given the small number of customers, but they will be more than sufficient for a small business. The number of potential niches is vast and there are numerous factors on whose basis we can segment and identify a market. The key question here will be how many potential consumers will this niche provide, as this will allow us to calculate the volume of business and hence, the volume of expenses that the company can afford.

Software offers more interesting possibilities than other tangible products in niche markets because of the absence of geographical barriers with the Internet. A niche detected in a given geographical area may be relatively easily extrapolated to other areas with similar needs or even be extended by itself, without the need for special efforts from the marketing company.

When we create products for niche markets, it is essential to know this particular environment very thoroughly. Besides technical skills, we need to have an excellent knowledge of the activities, priorities and modus operandi of the niche in question. Following Eric Raymond's rule, "Every good work of software starts by scratching a developer's personal itch", it is useful to start with a niche that we form part of, in order to better understand what needs and problems lie within it.

### Example 3.7. Knowledge of the environment

This is the case of the software developers' niche: it is a well-covered and exploited ground, since every programmer is both a creator and user with an intimate knowledge of the needs and problems of the sector.

Another important factor to consider is whether the product is going to be sold to corporate environments, small businesses or individuals.

Service companies should focus on corporate environments, governments and other organisations because private consumers rarely pay for software-based services. Product companies, however, may choose the prospective clients of the target market based on the features of their products and their business strategy. Corporate customers may be more attractive because they are more willing to pay for a software product and will also contribute to the generation of revenue through services.

In the corporate environment, companies will pay for a software product, but they will also pay for support, training, installation and integration of the product into their existing systems. Companies that purchase software generally pay 15% to 25% of the price of the licence in annual maintenance fees (Dan Woods,

Gautam Guliani, "Open source for the enterprise"). They also often seek custom developments to tailor the product to their specific needs. Thus, corporate customers will help software companies to generate revenue from services, giving them more guarantees of continuity. However, these new revenues will be more labour-intensive and the company will require careful management to ensure that the costs of providing the service do not exceed the income generated through it.

### Recommended reading

D. Woods; G. Guliani. 2005. *Open source for the enterprise: managing risks, reaping rewards*. . O'Reilly Media, Inc.

Moreover, support services are often offered for specific product versions, so maintaining services relationship can also help with the generation of revenue in the form of licences for successive versions: although clients have no interest in purchasing the new version, they will be obliged to do so because support for the older version is no longer provided.

The downside is that major corporate clients will be reluctant to hire the services of a small, new company. One of the key factors in hiring is the reputation and trust generated by the company providing the services, so smaller firms or those that have just started up will find it easier to obtain clients of a similar profile, i.e. small and medium-sized companies.

# Patterns of technology adoption and the "chasm"

## Important

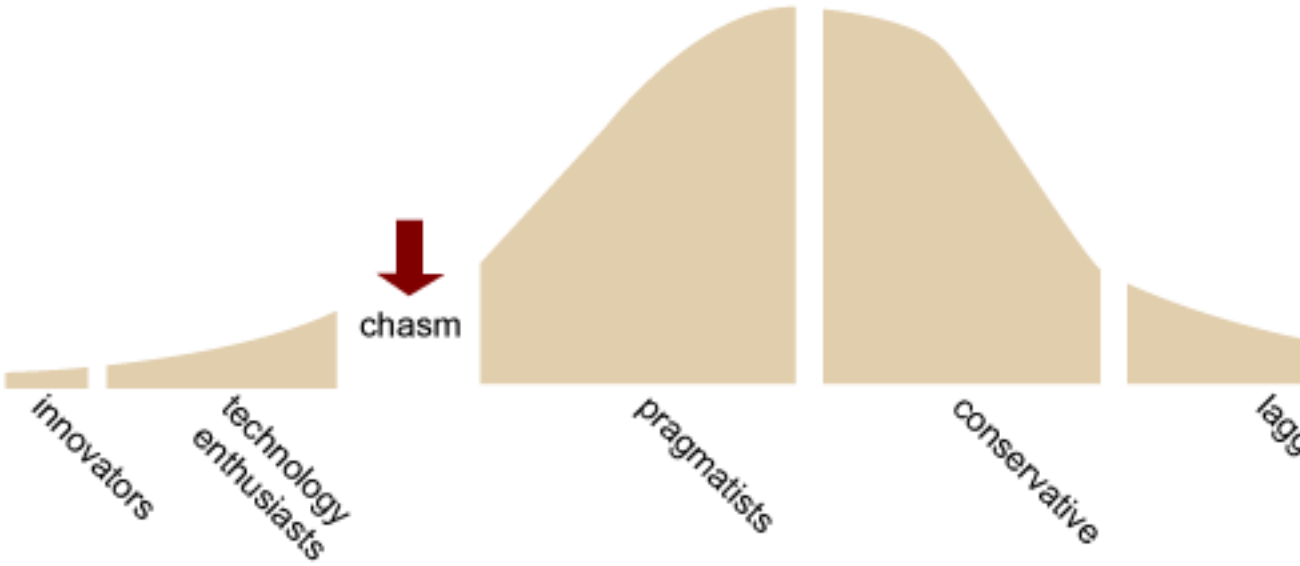
Detecting a market niche and creating a good product that meets the needs of the group of potential users is not enough to obtain acceptance. To introduce a new product or service, it is essential to take into account the patterns of technology adoption in a group of individuals.

Marketing books traditionally outline a model of adoption based on a Gaussian curve with four groups of users:

- **Innovators and early adopters:** these like technology and innovation. They often adopt a certain product simply because it is new.
- **Early majority:** these adopt a technology only if it helps them to solve a particular problem.
- **Late majority:** these try to avoid new technologies.
- **Laggards:** these are the last to try something new or may never get to try it.

The curve represents two key ideas: the two intermediate categories cover the vast majority of **potential customers**, and we can only attract the groups in order from left to right (early adopters will adopt it if the innovators have already done so, the early majorities if the innovators have, the late majorities if the early majorities have, and the laggards if the late majorities have).

Geoffrey Moore in his *Crossing the Chasm* renames these groups, calling them **technology enthusiasts**, **visionaries**, **pragmatists**, **conservatives** and **skeptics**, and argues that the theory is flawed because the transition between the enthusiasts and pragmatic majorities is not continuous and difficult to achieve. The early majorities will not adopt solutions that have not been extensively tested but they will adopt those that obtain good references from other pragmatists, so reaching them may sometimes seem like an impossible task. For Moore, there is a chasm between the two groups, so he redraws the curve as shown:




Innovators and technology enthusiasts have a **high tolerance for risk** and the flaws of the new technology because they already have significant technical skills. These users will adopt a technology on the basis of the pure functionality they reveal when seeking innovation. The early and late majorities (pragmatists and conservatives) have a **low risk tolerance** and will be interested in purchasing a product if it increases their productivity but only if it is highly stable and mature.

Thus, an innovative product can be a major success among innovators and technology enthusiasts, but if the creator wants to expand its customer base, it will need to launch a separate marketing campaign, focusing not on the specific features and enhancements of the product, but on generating confidence in it, describing success stories and previous implementations, and indicating numbers of users.

Gaining our first customers in the group of pragmatists and keeping them happy is essential but very difficult, given the vicious circle created: none will adopt a solution not previously tried by other pragmatists.

Confidence can be built by offering integral solutions, which include maintenance, support and training, to attract customers that are sensitive to the stability and user-friendliness of the product. The first customers in this group must be treated with care, with no time or money spared, as they will be the benchmark for the rest. Once we have gained a few benchmark pragmatists, attracting the rest will be a much easier task, and once the pragmatists have adopted the solution, the conservatives will follow without the need for great marketing efforts.

Concentrating on innovators and enthusiasts – on the assumption that, despite being a small potential market, it will be sufficient for a small business – can be dangerous because this group is inherently unstable and will abandon a product as soon as it ceases to be new.

This adoption curve will also mark the life cycle of the product, together with its dynamics of development and marketing practices. The marketing company needs to be clear on the stage it is at and who its customers are at that time, since each group is attracted by very different factors. While adding many new features and maintaining an evolving product will attract innovators, conservatives need the product simply to work in specific scenarios and for it to always do so in the same way. Every change will be a hurdle that they will only be prepared to face if it solves a problem they have.

# Function of the product: what to sell?

Careful consideration of the type of product to develop is very important. One of the questions we need to ask is whether the product is intended to be an industry leader, follower or a complementary product.

Although being the industry leader may seem more attractive at first, it may not be the most effective approach. When it detects a lack of functionality in a product with widespread adoption, a company has two options: develop its own version with the missing functionality and try to compete with the leader, or build an add-on to complement the possibilities of the leader.

The first option will prove very complex and can easily fail, as it requires a substantial investment not only in the new development but also in the marketing campaign and subsequent sales. In the second, besides the possibility of developing the product in less time, much of the marketing will have already been done by the leader, so it will be much easier to secure adoption of the add-on. Moreover, conservative users (the majority) will be much more willing to incorporate an add-on to a known and proven solution than to change technology and supplier. A common danger is that the leader may decide to incorporate the developed functionality into its core product, thus eliminating the need to purchase the add-on. In this respect, the relationship with the developer of the core product will be essential.

Consequently, it is important to define the role played by other companies active in the sector: which will be direct competitors, which will be partners and which, although in the same sector, will not compete with our product because they have a specific specialisation. By segmenting niches and offering differentiation, we can avoid direct competition from strong companies, and the existence of companies that produce related products or services may be an important factor in our success.

When positioning a product, it is also important to consider the platform that it is being developed for, i.e. which basic set of software will be required to run the product. Consider, for example, the choice of operating system and related technology with which the application will run. This decision will affect the definition of the niche market to be exploited and the type of customer it could be aimed at, but it will also be important for defining our relationship with allies and competitors.

An application designed to run on a particular platform will be a complementary application for that platform. If it is a software package already established on the market and widely accepted, we will also expand the potential market of our customers but reduce the chances of finding allies among the developers of the platform. The value of these platforms will be largely determined by the number and diversity of applications that can be run on it, so a company trying to establish itself as a platform leader will be very interested in the development of related applications and will hence be a more willing ally.

However, although it is more difficult, it may be better for the company to position itself as leader of a given sector. The question in this case will be whether to try and create a new product category for an untapped niche or whether to try and push out an existing product.

**Segmentation and potential customers**

For a modest company, the only possibility might be to segment the market until it finds a particular niche in which to position itself. It may be difficult to position oneself as leader in enterprise resource planning applications (ERP), but it could prove easier to develop an ERP for SMEs or for hotel and catering SMEs. Naturally, as we segment further, the competition will decrease, but so too will our potential customer base.

Topping a given market will undoubtedly generate advantages when it comes to positioning oneself as leader and defining the standards that this technology will be based on, but it offers no guarantees. The industry leader is not always the first company to develop a given technology. Sometimes, arriving first and attracting technology enthusiasts can give a false impression of success, since the product must reach the majorities before a company can become the leader. Subsequent strategic and technological decisions

will be critical in determining whether the company can capitalise on economies of scale on the demand side to position its product in the number one slot of its sector.

Breaking on to a market that already has a leader will take sales and marketing campaigns that are often outside the scope of recently formed companies. However, the use of a free software product, which competes with a price of zero, can be a sufficiently powerful disruptive agent. In future modules, we will see this and other strategies available to free software for competing on different markets.

# abstract

Software needs generate numerous business opportunities throughout the life cycle of the software, from development per se to related services such as installation, migration and user training.

Corporate positioning is key to identifying business opportunities:

- A service orientation provides a more stable economic framework over time.
- An orientation towards product development creates a product economy that is more difficult to maintain over longer periods.
- Hybrid models attempt to guarantee a balance between the above two models.
- The emergence of software as a service is a threat to more traditional models because it offers a more versatile variation for potential customers.

In addition, the exploitation of market segments that are close and familiar can help the business strategy of a new business and with the adaptation of the product to the patterns of technology adoption of the target market.

Lastly, it is also necessary to clearly establish the relationship between the business and its competitors and between its product and that of its competitors. These relationships may even encourage the introduction of the product on to the target market.

# Bibliography

## bibliography

Christensen, C. M.. 1997. *The innovator's dilemma*. . Harvard University Press < http://books.google.es/books?id=SIexi_qgq2gC > [Consulted in April 2009]

Christensen, C. M.; Raynor, M. E.. 2003. *The innovator's solution*. . Harvard University Press. < http://books.google.es/books?id=ZUsn9uIgkAUC > [Consulted in April 2009]

Cusumano, M.. 2004. *The Business of Software*. Free Press. Cambridge: Cambridge University Press. < http://books.google.com/books?id=7KAW-ToDnBAC&dq=the+business+of+software&hl=es > [Consulted in February 2009]

Daffara, C.. (March 2006). "Sustainability of FLOSS-based economic models".*II Open Source World Conference*. . Málaga. < http://www.cospa-project.org/Assets/resources/daffara-OSWC2.pdf > [Consulted in April 2009]

McKenna, R.; Moore, G. . 2006. *Crossing the chasm*. Capstone. < http://books.google.com/books?id=GTwFAQAACAAJ&dq=crossing+the+chasm&hl=es > [Consulted in February 2009]

Sink, E. . 2006. *Eric Sink on the Business of Software*. Apress. New Jersey: Princeton University Press < http://books.google.com/books?id=h5IQuengOGIC&dq=eric+sink+business+of+software > [Consulted in February 2009]

# Chapter 4. Business models with free software

Irene Fernández Monsalve

GNUFDL
2010-02-01

# Introduction

In the previous modules, we looked at the software market, the traditional types of company in the sector and the possibilities offered by free software in this framework. In this module, we will study the most common business models built around free software, together with some specific cases.

There can be no doubt that free software is emerging as a key element of new business models. After the bursting of the technology bubble (popularly known as the dot-com bubble) at the turn of the decade, free software has driven the creation of new companies in the technology sector, attracting increasing amounts of venture capital. In 2004, a total of $149 million was invested in 20 new companies. In 2006, this amount had risen to $475 million, distributed among 48 business initiatives.

### Recommended reading

*Open Source Business Conference (OSBC)*. For more information, see: Larry Augustin (2007). "A New Breed of P&L: the Open Source Business Financial Model". *Open Source Business Conference (OSBC)*.

http://www.osbc.com/live/images/13/presentation_dwn/A_New_ Breed_of_P_and_L.pdf [http://www.osbc.com/live/images/13/presentation_dwn/A_New_Breed_of_P_and_L.pdf]

Established and consolidated companies, such as Red Hat or MySQL, have been joined by a new generation of numerous companies whose strategies focus on the use and development of free software. Over the coming years, we will witness the real development of these new businesses and see whether their business model proves sustainable in the long run.

| First generation | Second generation | Third generation |
|---|---|---|
| **Publicly traded**:<br><br>Red Hat, Caldera (now SCO), VA Linux (now VA Software), Turbolinux<br><br>**Taken over**:<br><br>SUSE, Cygnus<br><br>**Other**:<br><br>LynuxWorks, Linuxcare (now Levanta), Sendmail | **Publicly traded**:<br><br>Trolltech, Sourcefire, Mandrakesoft (now Mandriva)<br><br>**Taken over**:<br><br>Conectiva, Lycoris, JBoss, Sleepycat, Ximian, Gluecode<br><br>**Other**:<br><br>MontaVista, MySQL, Zend | ActiveGrid, ActiveState, Alfresco, BitRock, Black Duck, CollabNet, Collax, Compiere, Covalent, DB4O, Digium, Exadel, eZ Systems, Fonality, Funambol, Groundwork, Hyperic, Ingres, Interface21, JasperSoft, Joomla, Laszlo Systems, Medsphere, Mozilla Corp, MuleSource, OpenBravo, OpenLogic, Open-Xchange, OTRS, Palamida, Pentaho, rPath, SnapLogic, Sourcelabs, Spikesource, SQLite, WebYog, SugarCRM, Talend, Terracotta, |

| First generation | Second generation | Third generation |
|---|---|---|
| | | Ubuntu/Canonical, Vyatta, WSO2, XenSource, Zenoss, Zimbra, Zmanda, etc. |

In this module, we will study some of the businesses listed in the table above, along with others that are still significant even though they do not attract much attention due to their small size. We will look at the advantages of the free software they exploit, the problems that they have encountered and how they have resolved them. We will also examine various taxonomies to characterise these models and try to identify diverse key factors that determine the operation of the company according to different authors.

# objectives

After completing this module, students should have achieved the following aims:

1. To understand the main classifications drawn up to date for free software models.
2. To know the current business models based on free software.
3. To understand the different mechanisms for revenue generation and differentiation exploited by these models.
4. To be capable of analysing how the different companies use free software to create a competitive advantage.

# Characterising business models with free software

When we talk about business models based on free software, we are often referring to the new and ingenious ways of earning income that are being implemented, since the traditional model, the selling of a proprietary product, is no longer so clear cut. Companies, in contrast to individuals, need to consider an important factor when they take part in a free software project: how to obtain the economic return that will justify their investment.

In previous modules, we saw how the idea that the income generated by software is directly related to its sale is not an accurate picture of the reality. Most software is developed internally and the sale of software is only the main source of income for a handful of companies. In most cases, it is necessary to offer complementary services to ensure the continuity of income and the survival of the business in harsh times.

Moreover, in the article by Perens that we looked at in the second module ("The Emerging Economic Paradigm of Open Source"), we saw that free software offers much better economic prospects (cost and risk) than the proprietary alternatives for companies that need to develop non-differentiating software.

### Recommended website

For more information on Perens:

http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1470/1385

In all events, this module will show how different companies manage the intellectual property of their products, also generating **mixed models** in an attempt to reconcile the advantages of free models with the generation of direct financial returns based on intellectual property. In this case, the choice of licence will largely determine the range of business models that a company can implement.

# Classifications according to different authors

In this section, we will study the various attempts to classify business models in literature, pausing to look at the factors that each author has considered crucial for the grouping of the different models. In addition to more theoretical approaches, we will look at those based on observing existing businesses in a more qualitative way and a quantitative methodology for the classification of business models in the context of the **FLOSSmetrics project**. Lastly, we will propose a taxonomy of our own that combines all of the proposals discussed.

# Hecker and Raymond classifications

One of the first authors to write about the business prospects of free software was Frank Hecker in 1998 with "Setting Up Shop: The Business of Open-Source Software". In his article, he takes four OpenSource.org categories and adds others, analysing them on the basis of:

### Recommended website

For more information, see:

http://hecker.org/writings/setting-up-shop

- Which companies implement this model?
- What types of licence are appropriate?
- What opportunities for differentiation does the model offer?
- What opportunities does the model offer to set prices based on perceived value rather than on actual costs?

The table below summarises this classification, adding another characterisation parameter, which, though not expressly mentioned by Hecker, is a key feature: how is the company revenue generated?

| **Model** | **Source of revenue** | **Type of licence** | **Opportunities for differentiation** | **Price opportunities based on perceived value vs. costs** | **Cases** |
|---|---|---|---|---|---|
| *Support sellers* | Sale of related services (covers all types of services, from custom developments to training, consulting, etc). | GPL | Quality, price, and simplifying and improving the user experience. | Limited.<br><br>Possible if it has a good reputation. | Cygnus Solutions<br><br>Red Hat<br><br>Caldera |
| *Loss leader* | Sale of other proprietary products | BSD or Mozilla | Based on the product. | Possible. | Sendmail<br><br>Netscape |
| Widget frosting | Sale of hardware | | Based on hardware: functionality, | Limited. The hardware pricing system | Corel<br><br>VA Linux |

| Model | Source of revenue | Type of licence | Opportunities for differentiation | Price opportunities based on perceived value vs. costs | Cases |
|---|---|---|---|---|---|
| | | | performance, flexibility, reliability, cost... | is typically based on costs. | |
| Accessorising | Sale of physical products (books, etc). | | Product quality (books, etc.) and loyalty from "pro-free software" users. | Limited. Brand reputation can allow prices to be raised slightly. | O'Reilly & Associates |
| Service enabler | Sale of on-line services provided by the program | GPL or Mozilla | Back-end attributes, creation of unique and useful services. | Possible if a unique and inimitable service is created. | Netscape |
| Sell it, free it | As a cyclical "loss leader" | BSD or Mozilla | Software functionality (while it remains closed). | Possible until the product becomes an interchangeable asset (at which point, it is released) | –hypothetical– |
| Brand licensing | Sale of name rights. The version co-exists with the "generic" branded version. | | Value added, for example, through additional validation and testing of the non-brand product. | | –hypothetical– |
| Software franchising | Sale of franchise and percentage of franchise revenue | | As a support-seller and brand licensing | Possible if it has a good reputation. | –hypothetical– |
| Hybrids (licences are neither free nor pure proprietary) | Limit code availability: sale of licences under certain conditions | | | | Trolltech<br><br>Qt |
| | User-based treatment on – sale to commercial users | | | | Open Group |
| | Treatment based on use – sale for commercial use, or sale for use on certain platforms | | | | Qt |

In *The Black Cauldron*, Eric S. Raymond also outlines the role of free software in business, focusing, among other aspects, on how free software affects the **"use value"** (**value as an intermediate product**) and **"sale value"** (**value as the end product**) of the software, proposing a taxonomy based on which of the two the company exploits.

### Recommended website

For more information, see:

http://catb.org/~esr/writings/magic-cauldron/

For Raymond, only sale value is affected by a free software model, so his classification describes models based on use value and models based on indirect sale value, in which free software makes the sale of another product or service viable:

- Models based on use value
  - Cost sharing (for example, Apache)
  - Risk sharing (for example, Cisco)
- Models based on indirect sale value
  - Loss-leader/market positioner
  - Widget frosting
  - Give away the recipe, open a restaurant
  - Accessorising
  - Free the future, sell the present
  - Free the software, sell the brand
  - Free the software, sell the content

As we can see, in the models based on indirect sale value, Raymond includes those of Hecker, plus one new one "Free the software, sell the content". In this model, the value lies in the information provided by the software platform, which is the information sold through subscriptions. The software is released, meaning that it can be carried over to different platforms, thus expanding the potential market of the real product: **the content**.

Though he proposes it only as a hypothetical model, Raymond anticipates the "**social website**" concepts and paradigm shift proposed by O'Reilly in his article "Open Source Paradigm Shift."

### Recommended website

For more information on "Open Source Paradigm Shift" see:

http://www.oreillynet.com/pub/a/oreilly/tim/articles/paradigmshift_0504.html

However, he does not recognise the role of the Internet as a platform or the subsequent "software as a service", considering that the value of releasing the software will lie in carrying it over to other platforms, thus contributing to its diffusion and market expansion.

# European Working Group on Libre Software

The business models presented by Hecker and Raymond are based on observation of companies that used free software as part of their business models, though they perhaps lack a degree of systematisation

and abstraction in their taxonomy. In its document "Free Software/Open Source: Information Society Opportunities for Europe?", the **European Working Group on Libre Software** (http://eu.conecta.it/paper/) makes an analysis based on how free software projects are funded rather than on the basis of their business models and regardless of whether the project is linked to a specific company:

- Public funding.
- Non-profit private financing.
- Financing for those who need improvements.
- Financing with related benefits (O'Reilly and Perl).
- Financing as internal investment.
- Other (bonuses, development cooperatives, use of markets to establish contact between clients and developers).

Its "Financing as internal investment" section, however, contains a classification of business models, which include, among others, the possibility of generating **revenue through services**, as a result of the competitive advantage afforded by being the main developers of a given software project.

| **Model** | **Differentiation** | **Revenue** | **Licences** | **Examples** |
|---|---|---|---|---|
| Better knowledge here | Better understanding of the product: must be the developer of the product or a collaborator. | Related services: custom developments, adaptations, installation, integration. | Free | LinuxCare (in its early days)<br><br>Alcove |
| Better knowledge here with constraints | Better understanding of the product: must be the developer of the product.<br><br>Part is kept proprietary. | Related services and sale of proprietary part. | Free and proprietary | Caldera<br><br>Ximian |
| Source of a free product | Producer, almost entirely free product. | Related services: custom developments, adaptations, installation, integration. | Free | Ximian<br><br>Zope Corporation |
| Source of a free product with constraints | Proprietary product in principle.<br><br>Subsequent release as a strategy to expand adoption and other advantages of free software. | Sale of commercial version. | Free and proprietary | Artofcode LLC<br><br>Ada Core Technologies |

| Model | Differentiation | Revenue | Licences | Examples |
|---|---|---|---|---|
| Special licences | Best knowledge here<br><br>Offer of proprietary version for customers who do not want GPL. | Sale of commercial version, and related services. | GPL and proprietary | Sleepycat |
| Sale of brand | Based on image and brand, allowing the product to be sold at a higher price. | Sale of distributions, and related services (including certification and training) | Free | Red Hat |

# Empirical studies

In "Business models in FLOSS-based companies", Carlo Daffara describes an empirical study of business models based on the use of free software, undertaken in the context of the **FLOSSmetrics project**. The study also examines how these models handle the marketing of their products and what licences they use.

### Recommended website

For more information, see:

http://opensource.mit.edu/papers/OSSEMP07-daffara.pdf

The study started out with 120 companies, of which it eliminated those not considered to be based on FLOSS (free, libre and open source software), and those that only allowed access to the code to non-commercial users or which did not allow redistribution. It also eliminated companies that, despite making important contributions to free software projects, do not base their core business model on it (such as IBM, HP and SUN).

It selected a set of characterising features, such as licensing, products and services offered (installation, integration, training, consulting, legal and technical certification), types of contract (subscription, licence, or per-incident) and self-referential literature offered on their websites and information on their relationship with the community. Lastly, the data were collected and all non-significant variables were eliminated to obtain the following characterising variables:

- Main revenue generator
  - Selection.
  - ITSC (installation/training/support/consulting). The different types of service are grouped together, since the study found that the companies offering one also tended to offer the others.
  - Subscriptions.
  - Licences.
- Licensing model

Applying cluster analysis to the companies characterised by these variables, the study obtained six basic business models, and a seventh group that was analysed separately:

1. **Twin licensing:**dual model of GPL and proprietary licence in order to sell to those who want to develop closed-source code based on the free product.

2. **Separate OSS and commercial products:** sale of commercial products based on a free one.

3. **"Badgeware":** brand protection; released products must keep original logo/authorship visible.

4. **Product specialists:** creation of a free product and sale of services relating to it.

5. **Platform providers:** selection, integration and support services, providing tried and tested platforms.

6. **Selection/consulting companies:** generic services and analysts do not generally contribute to the community, since the results of the analysis and consulting are kept private.

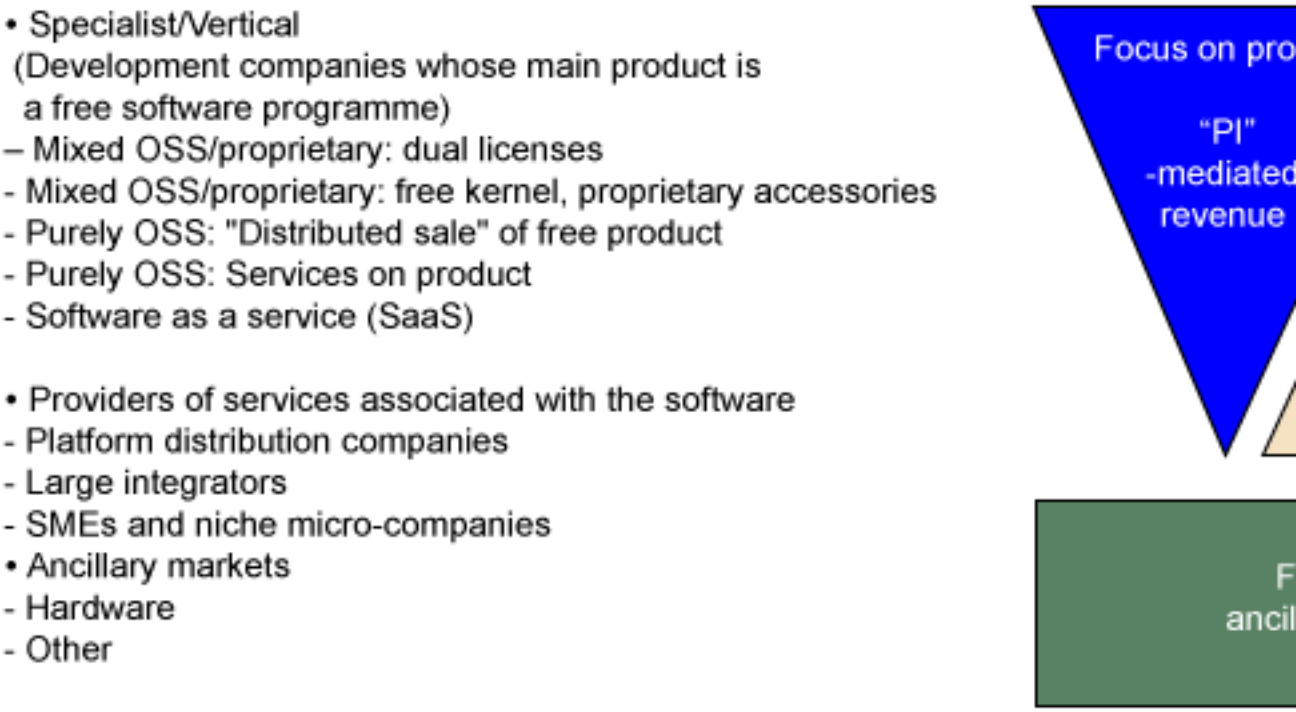
7. **Ancillary markets:** by way of example, SourceForge/OSTG generates most of its revenue from sales from its affiliate site, ThinkGeek. Although this model is not one of those characterised by the study (the limited number of cases in this category did not allow for extrapolation), it should not be underestimated as it is an important financing model.

The following table shows the results of the study.

| | | Main Licensing model | | | |
|---|---|---|---|---|---|
| | Company | twin licensing | OSS and commercial versions | Badgeware | Pure OSS |
| twin lic. | Funambol | • | | | |
| | Lustre | • | | | |
| | MuleSource | • | | | |
| | Mysql | • | | | |
| | OpenClovis | • | | | |
| | Pentaho | • | | | |
| | sleepycatdb | • | | | |
| Split OSS/ commercial releases | Adaptive Planning | | • | | |
| | Alterpoint | | • | | |
| | Altinity | | • | | |
| | Codeweaver (WINE) | | • | | |
| | Coupa | | • | | |
| | Digium (Asterisk) | | • | | |
| | Enormalism | | • | | |
| | EnterpriseDB | | • | | |
| | GreenPlum | | • | | |
| | GroundWork | | • | | |
| | Hyperic | | • | | |
| | JasperSoft | | • | | |
| | Knowlede Tree | | • | • | |
| | OpenCountry | | • | | |
| | Open-Xchange | | • | | |
| | NoMachine NX | | • | | |
| | rPath | | • | | |
| | Scalix | | • | | |
| | Sendmail | | • | | |
| | Smoothwall | | • | | |
| | Sourcefire (SNORT) | | • | | |
| | Splunk | | • | | |
| | SSLExplorer | | • | | |
| | SugarCRM | | • | • | |
| | TenderSystem | | • | • | |
| | VirtualBox | | • | | |
| | Vyalta | | • | | |
| | XenSource(Xen) | | • | | |
| | Zen(PHP) | | • | | |

# Proposed classification

The last classification analysed is interesting because it provides empirical data on real companies that currently focus their business model on free software. However, like Hecker, Daffara proposes a characterisation in isolation, rather than a taxonomy. We now propose a schema of our own to sort and incorporate the ideas we have analysed thus far, classifying the models by the degree to which their revenue is derived from the intellectual property rights over the software and by the extent to which they focus on the provision of products or services:




Our classification, like that of other authors, is based on source of revenue. Nonetheless, besides considering how the different companies recover their investment in free software development, it is also important to analyse how they exploit the advantages that a free development model can offer.

Business models are also characterised by their source of revenue, by the market they are aimed at, how they develop and market their products, and by how they relate to the competition. Hence, there is a cross-cutting issue affecting any business model that becomes particularly relevant with the use of free software: the concept of **coopetition**.

**Coopetition**

Among the other features differentiating free and proprietary software is the fact that the use of free software can enhance the quality of the services offered, thus helping to remove barriers to entry and sketching out a scenario of increased competition and effort for differentiation and specialisation, besides a distinct, open, cooperative competition in which companies will need to cooperate as well as compete if they wish to prosper. This business concept, which in some ways is replacing that of "winner takes all" in the context of a new network economy, is called coopetition.

## Important

**Coopetition**: cooperation between competing companies to seek win-win scenarios, either to enhance the value of the product or to expand the market.

### Recommended website

For more information, see:

Henry Chesbrough; Wim Vanhaverbeke; Joel West. "Open Innovation: researching a new paradigm"

http://www.openinnovation.net/Book/NewParadigm/Chapters/index.html

In this context, companies need to carefully examine their **economic ecosystem** – clients, providers, competitors and complementers – implementing strategies for the creation of new alliances and rethinking their traditional associations.

This concept is not unique to free software and has extended to other areas. Companies in the same industry can collaborate with one another to expand their markets, competing later when it comes to segmenting them.

**Intel**will invest considerable sums in expanding the microprocessor market, even though part of this investment will directly benefit its competitor, AMD. In this case, given Intel's dominant position, the percentage of its investment that will benefit others will be quite low.

Although coopetition is not exclusive to free software, it is highly significant in open-source development scenarios. It is inevitable that the competition will benefit from our investment, so it is necessary to find ways to turn this apparent disadvantage into a business advantage. Moreover, incorporating the users (clients) into the development process, involving them and encouraging their participation as allies, is also a feature of the free software development model.

To a large extent, the use of free software will also limit the possibility of becoming a monopoly and provide an anti-captivity guarantee. Again, a key question for any company becomes especially relevant in free software scenarios: how can we create value for a client while at the same time extracting some of this value for the company?

# Business models with free software

In this section, we will study each business model with specific examples. Note that these are not rigid models, but rather a diffuse continuous sequence. Many of the companies that we will mention combine several of the models, although we put them into categories for their systematic study.

## Specialist/vertical (a free application as the main product)

In this section, we include companies that produce free software as the promoters and/or leaders of specific projects. Their involvement with free software is thus very significant and one of the key aspects of their business strategy will be the management of the community and seizing of the opportunities for innovation, diffusion and volunteer work that it offers. In essence, these models have a free product for the community and a product or related service as their commercial offer, so the key to their success is often to strike a balance between the two. According to Marten Mickos, CEO of MySQL AB:

> FOSS companies will not work unless they serve equally those who want to spend time in order to save money, and those who want to spend money in order to save time".

These companies are the most common in Daffara's study, including the first four categories (twin licensing, OSS/proprietary versions, badgeware and product specialists). They equate to the product

companies we saw in module 3 of this subject, so their main problem will be how to recover the initial investment in development.

As we saw in the above classifications, a common strategy is to obtain revenue through proprietary licences, which are combined with free licences in different ways.

There are also **models that cyclically combine proprietary licences**, like Hecker's "loss leaders" and "sell it, free it". **The loss-leader concept is not unique to software**, being a widespread strategy in every sector of activity: a product is offered free of charge – or at such a low price that it entails losses for the supplier – as a way of attracting the attention of a large number of potential customers to whom the company intends to sell other items. Hence, both dual licence models and free products with proprietary extensions use a loss-leader strategy to some extent.

Besides promoting the sale of the related product, there are several benefits to adopting an open-source strategy of this nature in the software industry, such as helping to establish the technology as a *de facto* standard, attracting improvements and complements to make the product more appealing, generating sympathy in an audience that includes potential customers of the related product, and reducing the maintenance costs of the project.

We will now look in detail at the twin- or dual-licensing model and the model combining a core free product with proprietary accessories. We omit other models in which the main product is not free because they are really business models based on proprietary software: the code is only released as a complementary business strategy to enhance the position of the core proprietary product.

Daffara also lists several companies that carry out development projects with entirely free licences and earn their income from **ITCS**(installation/training/support/consulting). This group is perhaps one of those that can encompass the most different models, since its revenue source is a rather vague category. Hence, it is important to look closely at the markets they serve and their differentiation with equivalent products, in addition to best knowledge.

# Mixed models: dual licensing

## Important

This model is based on the distribution of a product under two different licences: a traditional proprietary licence and a restrictive free licence (GPL type). Thus, if somebody wishes to derive a work from it and redistribute the new work without the code, they can, but they must pay for a licence. Otherwise, all derivative works must be redistributed with the code.

Michael Olsen, manager of Sleepycat Software Inc., producers of BerkeleyDB, describes its dual-licensing model thus:

> "The Sleepycat open source license allows the use Berkeley DB [...] without cost, under the condition that if the software is used in an application that is later redistributed, the complete code of the application must be available, and must be able to be redistributed again freely under reasonable conditions. If you do not want to offer the source code of an application derived, you can buy a Sleepycat Software license."
>
> S. Comino; F. M. Manetti. "Dual licensing in open source markets". Available at: http://opensource.mit.edu/papers/dual_lic.pdf

This strategy is appropriate when a substantial proportion of the demand is generated by commercial users who need to embed the software in their own products. These customers use the product purchased as input for the production of new software, either as an end product or as part of a more complex technology

produced and sold by the commercial customer. Whether because they need to be able to sell their derived products under a traditional proprietary system or because the software they generate is a fundamental part of their differentiation, this customer will need to close the code it generates and must therefore pay to do so.

These models divide their users into two groups: the community – all users who are content with free licences, and use the product under these terms – and corporate clients sensitive to the reciprocal terms of free licences.

However, maintaining a community of people collaborating on the product can be problematic. On the one hand, if direct income is obtained through the product, this could affect the motivation of the volunteers who contribute without receiving anything in return. While on the other hand, the companies that implement it must formally obtain copyright assignment from the volunteers in order to avoid future problems from disgruntled employees claiming their share of revenue from licences for the product that they helped develop.

In practice, companies that base their model on dual licensing do not benefit greatly from the possibilities of external contributions to the development, obtaining only small-scale debugging and the odd patch from the community. The main development team is typically almost 100% dominated by company employees.

Another problem that can arise with these models is that their customers can build their own proprietary extensions without modifying the original code, so they can use the free licence version and have their add-ons as a separate and independent application.

These companies often combine the revenue generated from dual licensing with other activities such as the provision of services, which we will see later. Examples of this model include Funambol, MySQL, Sleepycat DB, and TrollTech/NOKIA.

## Example 4.1. The Funambol case

| | |
|---|---|
| **Company name** | Funambol Inc. |
| **Head office** | Redwood City (United States) |
| **Website** | www.funambol.com [http://www.funambol.com] |
| **Creation date** | 2001 |
| **No. of people employed in 2007** | 40 |
| **Turnover in 2007 (million)** | $4.8 |

**Funambol** is a US corporation engaged, as its motto states, in "mobile 2.0 messaging powered by open source". The company develops a mobile application server (providing push e-mail, address book and calendar, data synchronisation, and an applications server for mobile devices and PCs), together with a development platform for mobile applications, both developed under the name "Funambol".

It sells its code base under two licences: the AGPLv3 for its "Community Edition" and a commercial proprietary licence for its "Carrier Edition". It also combines this strategy with providing the additional functionality required for large-scale implementations of the closed version as well as services based on the "Carrier Edition".

In this case, Funambol chose the "Affero" GPL licence, which affords it extra protection against the commercial use of its applications in the form of software as a service (SaaS). As discussed in module three of this subject, the GPL allows the code to be modified without redistribution, provided that the application itself is not redistributed, as occurs with the provision of software as a service. The "Affero" GPL solves the problem of this void by requiring redistribution of the source code when the software functionality is offered under the "SaaS" model too.

The nature of the software makes it an ideal candidate for the dual-licensing model as it appeals to other companies seeking to develop closed applications on its platform, such as mobile telephony operators, device manufacturers and other software companies. Because of its use of the AGPL, companies that use Funambol as the basis of their "SaaS" offers must also pay if they do not wish to redistribute the code. Its customers include Vodafone, Earthlink and Computer Associates.

Funambol tries to fully exploit the needs of large corporate customers with the incorporation of additional functionality in its commercial version and services. To avoid the problems of the "free kernel + proprietary accessories" model that we shall see later, it makes sure that the closed functionality is only interesting in scenarios of large corporate implementations, so its free user community will not feel the need to develop this functionality by itself.

In "My Honest Dual Licensing", Fabrizio Capobianco, manager of Funambol, argues that the dual-licensing model is the most "honest" model in upholding the principles of free software development, making it more compatible with business need to generate profits.

However, as we saw earlier, defining a viable source of income does not guarantee the success of any company and the use of free software will allow us to implement qualitatively different strategies to those of a model based on proprietary software. Funambol is a case in point here, since the company had to refine its marketing practices and their target populations before hitting on a viable business model.

In its early days, Funambol tried to set up a classic software-vendor model around its free software product. The company developed Sync4j, which enabled developers to build applications for mobile devices under the "sometimes-connected" paradigm (the application can work offline, synchronising data when the connection is restored). It identified large companies and wireless operators as prospective clients which, due to large staff numbers and the increasing opportunities for mobility, would need to synchronise data between mobile devices and their corporate servers.

In order to reach these customers with its product, Funambol decided to pursue a proactive sales strategy with particular emphasis on marketing and its sales force, which tried to access the potential customers directly through telephone campaigns.

It met with very limited success. Funambol failed to meet its sales expectations because it found that large corporations were reluctant to deal with small new companies. In addition, the sales cycles were very drawn out and it soon became apparent that a much bigger sales and marketing team was required to maintain this strategy than Funambol could afford.

Funambol quickly realised that its problems were down to this active sales strategy, the traditional method in the world of proprietary software but a barrier to entry that only a handful manage to overcome: to access a group of potential customers consisting of large corporations, it is often necessary to have a large sales and marketing capacity, as well as a sufficient size and reputation to deliver the required confidence.

The use of free software allowed the company to reverse this strategy, focusing on a reactive type of marketing in response to initiative from the customer. In this new scenario, potential customers would seek out Funambol, leaving the company with the role of being attentive in order to identify them after contact.

The effectiveness of this strategy depended on a single factor: the number of downloads of its product. Once it had obtained enough downloads, it was able to identify the following typical sales cycle (much [illegible]):

1. The potential user accesses the Sync4j website for information on the product and technical [illegible]
2. [illegible]
3. [illegible]

The [illegible] case". Available at: http://oss2005.case.unibz.it/Papers/4.pdf)

As we saw earlier, maintaining a high number of product downloads. The cycle is continuous, generating more downloads by itself. Thus, after the initial effort, this mechanism gathers enough inertia to work virtually alone.

To do so, Funambol concentrated on **creating a community around its product**, focusing its marketing efforts on the users of its free version, both experts and those with fewer technical skills. Although this strategy is not directly oriented towards its revenue-generating customers, it proved much cheaper and more efficient.

The company focused on raising the profile of the product among developers, participating in development forums, mailing lists, specialist publications, conferences, creating partnerships with non-profit organisations that promote free software and establishing synergies with other well-established open-source products. For more inexperienced users, it had to ensure that the product was easy to install and that sufficient documentation was available on the website. When it began to focus on these last two factors, the company observed a substantial increase in the number of product downloads, thus setting in motion its sales-generating cycle.

### Recommended website

For more information, see:

http://www.funambol.com/blog/capo/2006/07/my-honest-dual-licensing.html

## Mixed models: free product kernel and proprietary accessories

In this model (Daffara's "Split OSS/commercial releases"), a program has two different versions: a free basic version and a proprietary commercial version based on the former but with additional functionality implemented through plug-ins or accessories. The free version must use an MPL or BSD type licence allowing the combination in order to create a closed product.

The main problem with this model lies in keeping the free product interesting enough without taking value away from the revenue-generating proprietary product. We also run the risk that the community formed around the product may decide to develop the functionality of the proprietary version on its own, making it difficult to generate revenue from sales.

In this model, we can distinguish between two classes of users: those who are willing to pay for a product with some additional features (medium and large companies), and those who are very sensitive to price, such as small businesses, micro-enterprises and private users. By combining free and proprietary versions, we obtain a more widespread adoption of the proposed solution without missing out on revenue capture from proprietary versions. As we saw in previous modules, in a "winner takes it all" scenario, common in software, strategies based on widespread adoption are very important.

Hence, it is based on the same user-segmentation principles as the dual-licensing model but is more at risk of losing the sympathy of the community, since it does not have access to the entire source code.

An example of this model is **Sendmail Inc.**, which sells an array of proprietary products around the sendmail open server. Other examples include Hyperic (IT Operations/Monitoring), SourceFire (SNORT commercial version), Zimbra/Yahoo (messaging, groupware) and XenSource/Citrix (virtualisation).

**Example 4.2. The Sendmail case**

| | |
|---|---|
| **Company name** | Sendmail, Inc. |
| **Head office** | Emeryville, CA.<br><br>(United States) |
| **Website** | www.sendmail.com [http://www.sendmail.com] |
| **Creation date** | 1997 |
| **No. of people employed in 2007** | 125 |
| **Turnover in 2007 (million)** | $23 |

When studying business models based on free software, we often think of corporations that decide to open up their code as a competitive advantage to expand their market share. Sendmail is an interesting case as this process occurs in reverse: with free, non-profit roots, the creation of a commercial initiative around the project is aimed not only at generating revenue from the development, but also to maintain the project's dominant position in its sector and to expand its user base.

Sendmail is a mail transfer agent (MTA) and one of the best known examples of projects born out of free software communities. In 1998, it was estimated that 80% of all e-mail traffic was sent through Sendmail. It is still the most popular MTA on the Internet, although it has lost some users to Microsoft Exchange Server, Exim and Postfix. Equally important is the long lifespan of the product, whose origins date back to developments started in the 1970s.

Eric Allman developed the first version of Sendmail at Berkeley University in the early 1980s on the basis of previous work on the Delivermail program and founded Sendmail, Inc. in 1997. The company strategy focused on selling additional Sendmail functionality in a proprietary format (e.g. user-friendly interfaces) in addition to providing complementary services. At the same time, the company made an effort to openly maintain the continuity of Sendmail's development by providing hosting services and human resources for its development.

When he set up the company, Allman expected not only to develop a business, but also to protect Sendmail's dominant position, which was being threatened by the emergence of proprietary formats that jeopardised the SMTP open standard. The company concentrated its efforts on the corporate environment, offering not only integration and support services, but also a product that was more responsive to its needs. The extensions created by the company provide graphical interfaces and ease of management, and are marketed in proprietary formats.

"Sendmail, Inc. develops commercial products and services for ISPs and enterprises for whom email is mission critical, while continuing to drive innovation and standards through Open Source software development."

Sendmail, Inc

We can consider the creation of Sendmail, Inc. to have been the necessary step to cross the "chasm" and guarantee the product's adoption by the pragmatic and conservative majorities. Nonetheless, for Allman, it was important to maintain the original functionality of free Sendmail, so Sendmail Consortium was set up as a non-profit entity to develop the free version. In this way, it can capitalise on the advantages of an open development model, such as contributions, cost-cutting, product innovation and evolution.

Allman thus took advantage of "the chasm" to sell proprietary extensions to his product without the danger of forking his project. Following Moore's model, the community around the free Sendmail project consists of innovators and technology enthusiasts interested in the raw functionality and new proposals. Business customers, however, are pragmatists and conservatives with very different needs and aims. The proprietary extensions, which focus on the functionality of the product packaging and finish (ease of use, graphic interfaces, stability, etc)., are not only uninteresting to innovators, they may even seem unnecessary. The presence of this chasm between the interests of the community and commercial customers allows for the co-existence of the core free version and the widespread proprietary version without the risk of forks, since the community has no interest in the extensions on the other side of the chasm.

# Free models: "distributed sale" of the product

It is commonly assumed that licensing a product in free format leads to loss of opportunity for earning direct revenue from the intellectual property rights over it, creating the need to exploit complementary products or services.

However, choosing a free licence for a project does not necessarily mean forgoing the possibility of obtaining revenue directly from this product. The widespread idea that nobody will pay for something they can obtain for free does not paint a true picture of reality. Many people are willing to pay a small sum for a work that they value if they think that this money will go to the original authors. If a project is successful enough, it may receive small contributions from a lot of people, perhaps even managing to fund its creation in the same way that a street artist does not charge admission but can raise enough to make his or her investment in time and effort worthwhile. This is the idea behind the*The Street Performer Protocol and Digital Copyrights*, by John Kelsey and Bruce Schneier, which proposes a distributed funding mechanism for digital works in which the author does not complete his/her work until sufficient funding has been collected.

### Recommended website

http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/673/583

Different mechanisms have been described and implemented for structuring this direct, distributed funding in the context of software development, from grants and bounties to the creation of on-line markets that bring together developers and prospective clients, based on a bonus scheme similar to that described by Chris Rasch in*The Wall Street Performer Protocol*.

### Recommended website

http://www.firstmonday.org/issues/issue6_6/rasch/index.html

Donations are the most straightforward mechanism for this type of financing, but too unstable for creators, who need the security of an income before they invest their time. In bonus and bounty systems, the people interested in a specific functionality offer a reward for it to be implemented. When the total reward – which various people can contribute to – reaches a sufficient sum for a developer, he or she can offer to do it and is paid once it is finished. Some of these systems rely on the trust between the development team and the users, and have no payment guarantees, while others propose the establishment of some form of neutral intermediary.

## The Cherokee server

This server decided to implement a bounty system with the primary aim of attracting new developers to the project. Besides rewarding effort, providing a financial incentive would attract more people to the development community and encourage the growth of the project.

### Example 4.3. Virtual markets

Several attempts have been made to create "virtual software markets" based on this type of funding. Some of those currently operating include BountyCounty (http://bountycounty.org/) MicroPledge (http://micropledge.com/) and BountySource (https://www.bountysource.com/).

The key to success in these scenarios may lie more with the payment facilities offered than with the willingness of users to pay:

> "Most people are happy to pay a tiny extra bit on top of some larger amount, if they have their wallet out already and think it's for good reason. When people fail to make small, voluntary donations to a cause they like, it's more often due to the inconvenience (writing a check, putting it in the mail, etc), than the money.
>
> (Karl Fogel. "The Promise of a Post-Copyright World". Available at: http://www.questioncopyright.org/promise)

Although many projects implement these ideas to obtain additional funding, it is difficult to identify corporate scenarios where the bulk of the revenue is obtained through these mechanisms.

Firstly, in the context of software, this type of funding can be more difficult to obtain because of the absence of a strong identification with and sympathy for the authors, which does exist with other creative works.

Secondly, this model is likely to be more successful if it is a non-profit free software project composed entirely of volunteers, which will arouse the sympathies of its users more easily. A company wishing to use it successfully will no doubt have to obtain prior acknowledgement through transparency and trust, proving that profit-making is not the be all and end all and that the project will have an impact on the common good (we will look later at business models based on these principles).

These systems have a more direct economic model, eliminate intermediaries and ensure greater proximity between users and developers. In one sense, they could be considered the natural way to fund a free software project: just as volunteers contribute to varying degrees and in different aspects of the software development cycle, so too can users form part of the project by making a financial contribution in line with their possibilities and interests.

## Free product plus associated services

Companies in this category implement a strategy of the type "best knowledge here" and "best code here", developing a free product and offering services for it as a means of generating revenue.

This sections encompasses both the product specialists and badgeware of Daffara's study, since they both represent the same business model. Moreover, although badgeware licences include an additional assignment constraint, they maintain the essential characteristics of openness and freedom of knowledge and can generate the same benefits through their development communities as those that use licences without this constraint. The companies constituting examples of badgeware probably also seek to launch some sort of brand strategy, so they place special importance on assignment when redistributing the products they generate.

This model has a number of problems, such as few barriers to entry to the business – any company can gain knowledge of the product and offer services – and problems obtaining support contracts as client companies may prefer to continue with their regular service or consulting companies or to hire providers that offer support for their entire new technology infrastructure and not just for a specific product.

Another common problem faced by these models for generating revenue from services is that of innovators and enthusiasts: when a new product comes on to the market, its early users are often people with technical skills that will not contract support services for it, preferring to acquire the necessary knowledge for themselves. This model then will need to offer an extended product and transmit reliability in order to reach a potential market that will pay for services relating to the product.

The success of this type of business model is questioned by some authors (like Perens). Nonetheless, there are many companies based on this model that have attracted large sums of venture capital. For a more sustainable business model, however, they will need to address the problems mentioned above.

The models of vertical service provider specialists include Alfresco (content management), Compiere (ERP, CRM), vTiger and Openbravo.

| | |
|---|---|
| **Company name** | Openbravo, S. L. |
| **Head office** | Pamplona (Spain) |
| **Website** | www.openbravo.com |
| **Creation date** | 2001 |
| **No. of people employed in 2007** | 26 to 50 |
| **Turnover in 2007** | Up to €300,000 |

## Example 4.4. The Openbravo case

**Openbravo** is an interesting example of this type of model. The company, founded in 2001, develops two free applications for SMEs – OpenbravoERP (enterprise resource planning) and OpenbravoPOS (point of sale) – which seek to meet the needs of management and planning and of point of sale terminals for small and medium enterprises, respectively. The code was published in 2006 and is currently among the most active projects in the SourceForge ranking.

Its business strategy is geared towards becoming a leading product in the industry and making OpenbravoERP the benchmark for management software among SMEs. To achieve this, the company is exploiting the possibilities of free software to the maximum through careful community management and application of the competition concept.

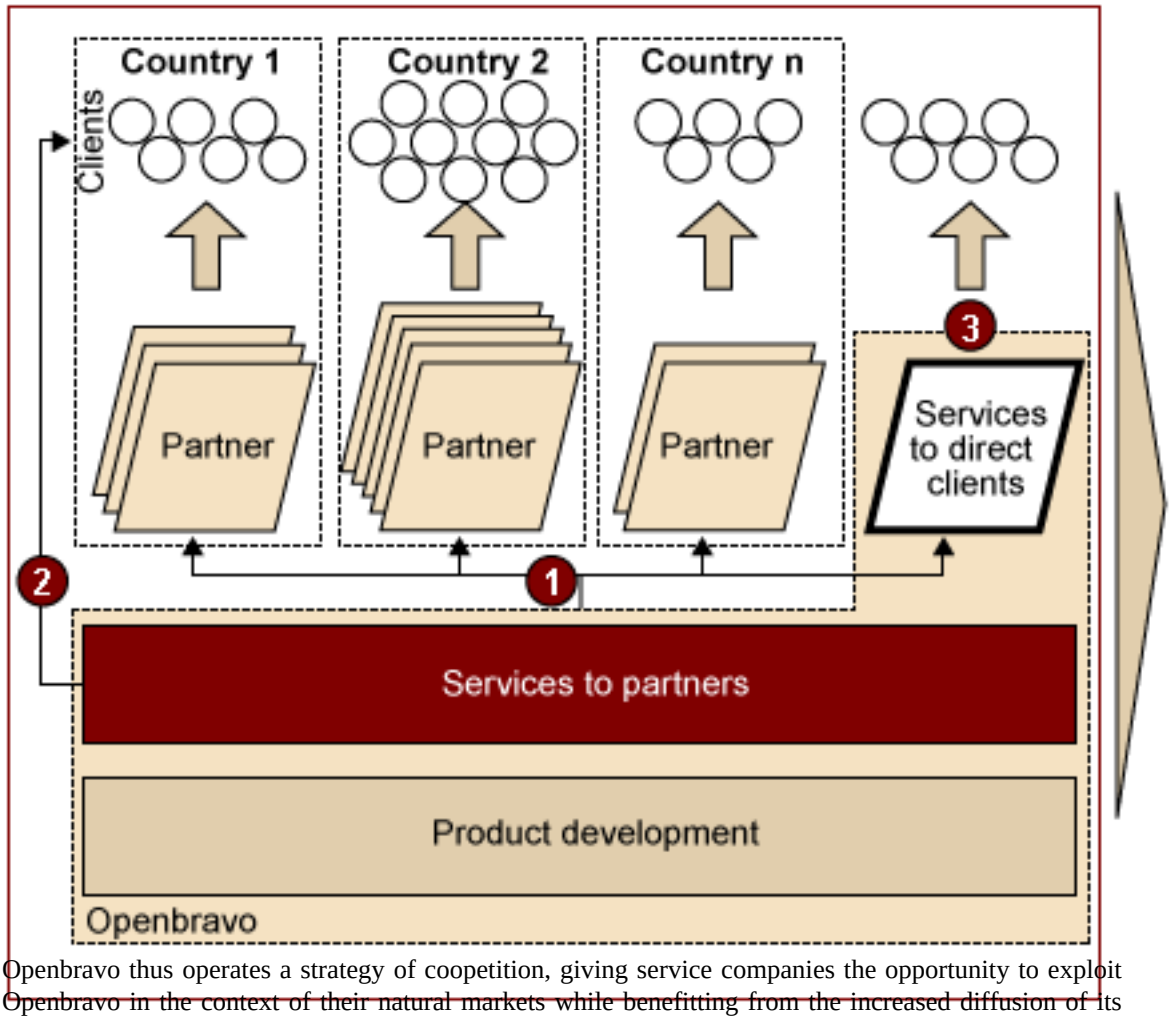
As we saw with Pulambol, Openbravo observed that, alone, it did not have the capacity to disseminate and distribute its product among potential users. Although OpenbravoERP and OpenbravoPOS are aimed at SMEs rather than large corporations, in order to achieve its strategic aims of becoming the sector leader, the product had to reach countless small and medium businesses worldwide.

In addition to the size requirements for conducting a marketing campaign of this scale, the company was also aware of the potential difficulties of competing to provide services directly to end users, who might prefer local businesses or ones offering integral solutions and not just companies dealing with a single product.

To overcome these barriers, Openbravo positioned these companies as collaborators rather than competitors. Thus, it admits that, simply because it developed the product, this does not necessarily mean that it is the best company for providing related services to end users. Its mission was to create a good product that could expand markets, generating new revenue opportunities for IT services companies, which could complete its offer with OpenbravoERP and OpenbravoPOS.

Thus Openbravo defines the provision of services to other IT services companies – intermediaries between Openbravo and the end users, as we have mentioned – that thus form a network of partners that carry out the tasks of implantation of OpenbravoERP and OpenbravoPOS in SMEs.

Openbravo offers its partners various services (support, training), by implementing a pyramidal system of consulting similar to that described in module 3 of this subject, as well as conveying reliability and trustworthiness. As they are supported by the product developers, they can exploit the strategy of "best knowledge here" and "best code here" on their markets.




Openbravo thus operates a strategy of coopetition, giving service companies the opportunity to exploit Openbravo in the context of their natural markets while benefitting from the increased diffusion of its product, and obtains revenue directly from its partners. Thus far, it has been considerably successful with this strategy and currently has eighty-five partners around the world.

## Software as a service

Companies that develop a product can also exploit it through the paradigm of software as a service. Instead of offering installation and support services, the company is responsible for all hardware and software infrastructure, offering functionality directly through the Internet. The recurring revenue generated takes the form of service subscriptions.

### Example 4.5. Collabnet: software as a service

A good example of this type of model can be seen in **CollabNet**, which provides services for collaborative software development (version control, issue tracking, communication, etc.), generated, among others, through the Subversion version control platform. In this case, in addition to keeping the source code open, the company spends a lot of effort on maintenance of the community, so that its work on the project is merely a contribution – albeit a large one – within a free community. Other examples of companies that market their products according to the "software as a service" model include SugarCRM, SocialText and JasperSoft.

With the "software as a service" format, these companies will not come across any more difficulties generating revenue than their proprietary equivalents, since the sales in this case are not derived from the copyright on the product. The fact that a client can download, install, configure, host and maintain the application will be more a tool for marketing and distribution than a loss of income. As noted earlier, corporate clients are willing to pay for having their problems solved.

Nonetheless, releasing all of the code creates problems with differentiation and opportunities for the entry of competitors. Any company with a sufficient technical capacity and infrastructure could offer a similar service if the code were available. In the light of this problem, the company that developed the product could base its differentiation on "best knowledge here" and "best code here" to gain the sympathy of the community. In addition, if its competitors also chose to contribute to the development, it could set up coopetition mechanisms, collaborating to expand the market and segmenting it later according to specialisation.

Like the mixed OSS/proprietary strategies we saw earlier, some companies in this category will implement solutions incorporating some form of restriction on their code, mainly by keeping a small section of the code closed, which will form the basis of their differentiation.

# Services associated with free software

Considering the services associated with free software, there are many possible businesses because, in general, any services model based on proprietary software (such as those discussed in module 3) can be extrapolated to free software in a fairly direct way. All of the steps described in the chain of creating and implementing a technology solution are viable in the context of open applications. However, the use of free software extends the possibilities and differentiation factors of business models focused on services.

One of its basic differentiating principles is the **absence of licensing costs**, giving it a clear competitive advantage over proprietary solutions. Nonetheless, in order to take advantage of this factor, it is important for the proposed solution to be cheaper in the long run (considering the "total cost of ownership") and to provide a standard of quality at least equivalent to its proprietary competitors. It is also crucial for companies offering free software services to be more attractive to customers by reducing the possibility of lock-in situations: these providers cannot rely on continued income in a situation with captive customers; instead, they must be based on the **continued provision of quality services**.

On the other hand, just because a software is free, this does not mean that it will be accessible to everybody. The market for service companies will not diminish due to the availability of free applications or those at no cost, since the task of selection, installation, training and support will always be necessary in corporate environments, and it will be more interesting if the licensing budget is spent on improving service.

As a rule, these types of company are involved in various projects, though not intensively in any. Some will contribute, as is the case of platform distributors, with debugging, especially in areas of customer interest, and on the tasks of integration and ensuring compatibility between different applications. Others, such as those that focus on consulting and selection (with no capacity for development), will not contribute to the projects on which they are based, since their work is usually kept private and will not be visible to the public. In these cases, however, a return can be obtained in the form of the promotion and adoption of the solution on which they work.

There is a vast range of possible models in this category (differentiation with respect to size, solution segmentation – horizontal or vertical – industry segmentation, specialisation in a particular service: custom development, selection, consulting, integration, training, etc.), and most companies will offer a combination of the possible services. First of all, we will look at the special features of free software in the different stages of implementation of a technology solution, before turning to the specific typologies of business models, which group certain services in a particular way.

**Custom developments**

Free software offers companies a compromise on the question of "**to buy or to develop**". These companies can start with a free standard product and, either internally or through a development company, build the necessary adaptations to suit their needs. Both the service companies that we will look at now and the product-oriented ones we saw previously will receive offers to perform this type of customisation. However, making these adaptations privately, without trying to incorporate them into the master project, can be problematic when it comes to maintaining compatibility between the adaptations and subsequent versions. Hence, working with the community, designing the new features so that they can appeal to more people, and incorporating them into the main code of the project will save a lot of work and complications.

**Selection**

The presence of a wide range of applications within the (economic) scope of any company makes selection a critical task. Not only will it be necessary to find products that better suit the needs of the client company, they must also evaluate the health of certain projects, the pace of debugging and new releases, and their stability. For corporate environments, a project with a lot of movement and a rapid rate of adoption of improvements may not be the best, since a stable product that will not change significantly over time may be more appropriate.

**Installation and integration**

## Additional reading

D. Woods; G. Guliani. 2005. *Open Source for the Enterprise: Managing Risks, Reaping Rewards*. .

Although this phase also generates needs in commercial environments, free software has a special business opportunity in this field: its lack of packaging and final finish. In*Open Source for the Enterprise*, Woods and Guliani allude to the concept of "productisation" as one of the main shortcomings of free software for achieving widespread adoption. The term refers to the degree to which the application has been packaged and prepared for end users, with the development of automatic installers, graphical configuration interfaces and sufficiently detailed documentation which, in short, allow for its installation and use by inexperienced users.

As a general rule, commercial software comes more packaged and finished than the free software developed on a voluntary basis. The installation scripts, administrative interfaces and documentation are usually more complete for a proprietary commercial product than for a free software product of the same age. While this lack of product completion is irrelevant for technology enthusiasts – indeed, it can even be more attractive because the adaptation and administration can be more direct and personal – to cross the chasm and reach the corporate client, free software must have a higher degree of packing and finishing. According to Woods and Guliani:

> "A broad oversimplification about open source versus commercial software is that open source represents primarily an investment of time, and commercial software represents primarily an investment of money. Any organization setting out to use open source must set aside some time for research and experimentation. "
>
> Dan Woods and Gautam Guliani. "Open source for the enterprise"

This time investment for completing an open-source application or selection of applications offers an important business opportunity both for platform integrators and developers. Hence, a good symbiosis could be established between the private sector and non-profit free software projects in which the investment would be spent on more monotonous work, leaving the more creative and innovative work to the volunteer community while also allowing the simultaneous creation of more mature products that are more likely to attain a high level of adoption.

Furthermore, both the modularity of free software and its coexistence with proprietary systems can generate serious compatibility problems, which require painstaking integration. The generalisation of standards will be beneficial for minimising the adverse effects of combining different software elements.

**Technical certification**

The inherent features of the finish of free software also allow for the possibility of certification by integrators and external consultants. This can take two forms: certification of compliance with international standards or certification of suitability for specific technology environments. The certifier provides assurance that the package meets a series of requirements and is legally responsible for their compliance.

Hence, the certifier provides an intermediary responsible for a set of solutions, an essential factor for many new technology departments of software consumer companies. Often, when an information technology department arranges support and maintenance, it is not only hiring a method of resolving incidents, but also a person or company to which it can attribute the problems or failures that may arise. The decision to adopt a particular free software solution without intermediaries to offer guarantees puts all of the burden of success or failure on the department itself, which may prefer for the intermediary to assume this responsibility.

## Additional reading

S. Sieber; J. Valor. 2005. *Criterios de adopción de las tecnologías de información y comunicación*. . IESE.

<www.iese.edu/en/files/6_15211.pdf>

**Training**

Training can be a source of easy income. In addition to the fact that the open development model makes the information on a product available to everybody, most free software projects lack formal training programs, meaning that anyone can enter the business. Many established companies whose business is training have added free software programs to their offer.

**Support and maintenance**

We have already seen how support and maintenance services are an important source of revenue for companies engaged in the development of a free product, but they also form part of the offer of companies that only provide horizontal services, as we shall see below.

As we said earlier, the possible range of service companies is vast, with models being developed on the basis of specialisation in certain services, a type of applications, local market or large scale, etc. We will study three typologies in detail. Firstly, platform distributors, as they were one of the first business models implemented with free software and are fairly representative of a number of major companies in the sector. Secondly, we have chosen two examples at either end of the scale: large integrators and small niche micro-enterprises. Between the two, we have the other possible business models, which focus on the provision of services.

# Platform distribution companies

The activity of this type of company is concentrated on the integration and selection of components to generate a **comprehensive software solution**. The diversity of applications and results generated by the free software development model requires integrated teams to give cohesion and ensure the compatibility of the parts. This has given rise to the emergence of different distributions developed by different actors. This activity is also an obvious potential business model.

Platform distribution companies use a similar model to application development companies and service providers, but the **selection and integration of a broad product base**, as opposed to development, lies at the crux of its work.

## Important

Companies using this model generate and distribute integrated software packages, mainly for corporate customers. The platform generated is the company's core product, which generates one major problem: product differentiation is very difficult because it is freely accessible.

Besides distributing software under a traditional model through the sale of packaged CDs, these companies often supplement their offer with services such as installation and quality support, often through a subscription system. Their added-value is based on reliability and trust, conveyed by the brand that represents them. They offer to fill in the gaps that a free software product may have for corporate environments, which seek an appropriate, stable and reliable solution – even at the cost of features and performance.

Thus, their potential customers will be medium and large enterprises, which require maturity and stability, professional support and a viable ecosystem of solutions. The investment in software is amortised over five years, so a company that is going to invest in software will need to know that – at least for this period – it will have support for these products. Given the extra costs associated with switching from one technology solution to another, having support that lasts beyond the amortisation period is highly desirable.

Hence, generating trust is fundamental to their business strategy and must include the development of a brand that conveys added reliability to a free software product. Given that their business model is based on a product freely accessible to anyone, these companies seek to develop a strong brand as a differentiating factor that will allow them to gain market shares over the same or very similar products.

Although these companies do not usually focus on the development of specific applications, they do often contribute to projects that they draw on by debugging, and develop new products only when necessary to expand the market for their product.

### Example 4.6. Red Hat

The archetypal example of a platform distributor is Red Hat, Inc., and this is also the model followed by Novell with SUSE, Canonical with Ubuntu, and Caldera Systems with Caldera Linux.

**New distributors**

New distribution companies have recently emerged, offering more specialised software bundles for more limited markets. SourceLabs, SpikeSource and Wild Open Source are examples of such initiatives. SourceLabs, for example, offers certified collections of software usually used together, such as Linux, Apache, PHP and MySQL. Wild Open Source, on the other hand, customises distributions for use in high-performance contexts or embedded systems. Along with the certified bundle, the companies offer maintenance and support services for their selection, just like traditional subscription companies.

The main challenge for this type of company will be to define software collections that are wide-ranging enough to maintain a sufficient customer base while being able to provide support for all elements in the bundle.

### Example 4.7. The SpikeSource case

| | |
|---|---|
| **Company name** | Spikesource, Inc. |
| **Head office** | Redwood City, CA. (United States) |
| **Website** | www.spikesource.com [http://www.spikesource.com] |
| **Creation date** | 2003 |
| **No. of people employed in 2006** | 80 |
| **Turnover in 2007 (million)** | |

**SpikeSource** is a representative example of the business potential generated by the lack of finish of free software products. Set up in 2003 by one of the most important venture capital firms of the Internet boom – Kleiner Perkins Caufield & Byers – SpikeSource launched its first products in April 2005. In October 2006, the company announced its expansion into Europe through a network of local solution providers and technology partners.

Murugan Pal, founder, summarises the company's activity as follows:

SpikeSource's goal is to facilitate the adoption of open source software in the enterprise through testing, certification and support services. We innovate, automate and optimize advanced testing techniques as part of our core competency."

(Murugan Pal. "Participatory Testing: The SpikeSource Approach". http://www.oreillynet.com/pub/a/network/2005/04/07/spikesource.html)

As differentiating factors with classical integrated solution distributors like Red Hat, the company highlights its efforts to promote testing automation and its combination of specific applications that can be installed on different platforms and operating systems. It includes versions for different operating systems, both free and proprietary, and incorporates closed software in some products.

In addition to its bundles, such as SpikeWAMP-1.4, which includes the latest versions of PHP, MySQL and Apache (for Windows installation), and "SuiteTwo", which integrates a wide range of embedded collaborative applications, and "Web 2.0" features, it recently launched a platform for developers on which they can test and integrate their applications, thus obtaining SpikeSource certification and a better software finish (productisation) as a result.

The work of this type of company can be very positive in increasing visibility and promoting the adoption of free software solutions, and SpikeSource has tapped into this. The company makes great efforts to show that its work benefits the free software community – and that it does not simply exploit it – by including well-known figures from the world of free software, such as Brian Behlendorf and Larry Rosen, on its advisory committee as endorsements. It has also developed a website for developers (http://developer.spikesource.com), where it offers its automated testing services for integration and compatibility on various platforms.

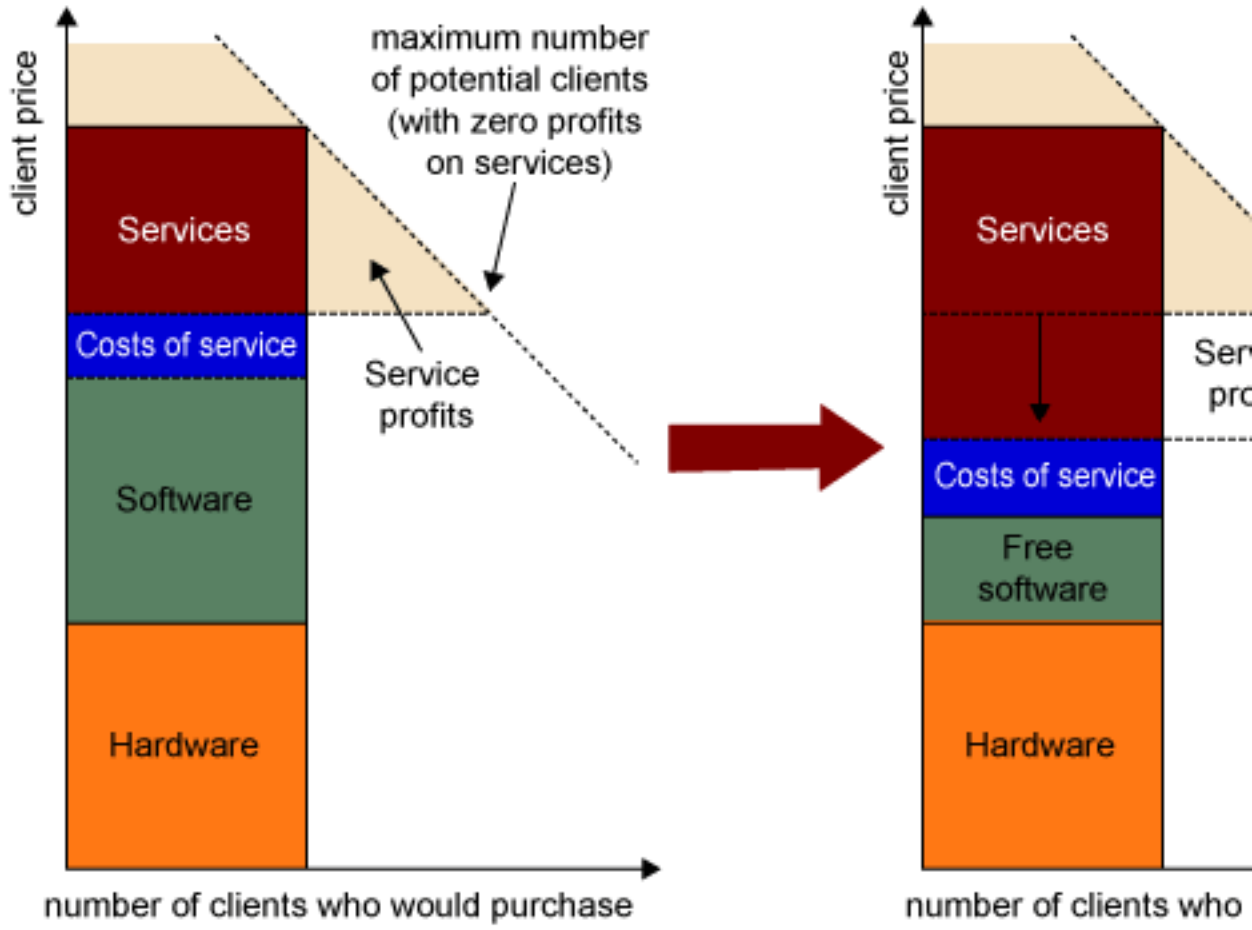
Nonetheless, the **automation software** used by the company combines parts that have been released with parts that remain closed. In this case, reserving a portion of the code is a strategy to protect its differentiation and keep competition from comparable services at bay. This decision reveals that rather than losing revenue from licensing (which, as we have repeatedly seen in this subject is not a real obstacle), the use of free software affects the company's possibilities of differentiation – and hence, business. In the case of SpikeSource, the effort invested in its testing applications will be rewarded not by the sale of licences for this software but by the protection of its differentiating factor from other companies offering similar services.

## Large integrators

Large systems integrators or solution generators are one of the types of company that stand to gain the most by basing their business on free software, given the direct cost savings, and the subsequent possibility of reaching more customers.

Clients usually look for companies that can provide solutions to an information and communication technology (ICT) problem and are not concerned with implementation details. A complete solution will combine hardware, software and services, making the process easier for the customer, who need only contact a company to solve its ICT problems and not have to worry about compatibility between providers. Therefore, everything that the company can save on software costs by using free software can be transferred to the costs of services, which will enhance the solution. The company can slash prices to increase its potential number of customers, or simply enhance its profitability. This type of large integrator, which generally works on complex projects, can maintain its prices due to the barriers to the entry of other competitors.

The figure below illustrates this situation, outlining the demand curve for comprehensive solutions and provider costs.




There are many consulting and selection firms, including Ayamon, Enomaly, Navica, OpenLogic, Optaros and X-tend. Large integrators include IBM, Sun and HP.

### Example 4.8. The IBM case

| | |
|---|---|
| **Company name** | IBM |
| **Head office** | Armonk, NY<br><br>(United States) |
| **Website** | www.ibm.com [http://www.ibm.com] |
| **Creation date** | Its origins date back to 1896. In 1924, it changed its name to IBM |
| **No. of people employed in 2007** | 394,540 |
| **Turnover in 2007 (million)** | $91,423 |

Twenty years ago, **IBM**was one of the strongest advocates of intellectual property rights for software. Its argument was that without strong copyright protection, there would be no incentives for companies to invest in software development.

Now, although it has retained the bulk of its proprietary software, IBM has launched major campaigns in support of free software, offering considerable financial contributions to the development of Linux and other applications, and the release of applications such as the Eclipse development platform and part of its AIX operating system.

IBM's current business model focuses on the **sale of high-end hardware**, proprietary software on Linux and the provision of **integration services for corporate clients**. Although IBM has been one of the world's leading software manufacturers, its programs have usually been marketed as a combined solution with its own hardware. As a result, the company has little to lose from lack of differentiation in the software that it uses: given the barriers to competitor entry in mainframes, the use of low-cost software allows the company to cut its prices and extend its range of customers without undergoing a loss of differentiation that would significantly increase its competition.

Thus, its use of Linux allows IBM to offer a lower overall price for its hardware and services, while also providing a common platform on which to build and sell applications and special services. Along these lines, we can also mention the savings made by the company through the use of an operating system with wide prior adoption – in marketing, distribution and sales terms – as well as the reduction in risk and investment in development. Moreover, the public image benefits obtained have also been significant.

Naturally, IBM's free software activity involves a more complex strategy that affords it a better competitive position on several fronts. From strategies based on the use of free software to enhance the marketing of its proprietary products (such as "loss leaders" and free kernel plus proprietary accessories) to gaining a better position than other big software providers.

The use of free software has given IBM more independence than other large companies, such as Microsoft, and a better position over direct competitors like Sun. The latter has, for a long time, based its business strategy on the combined sale of hardware plus "better than average" operating systems and would therefore have more to lose in the event of cost-cutting and the presence of equivalent low-cost software.

## Software services: small and micro-enterprises

### Important

Another basic phenomenon sparked by free software is the **transfer of knowledge and technology**. Investments in training, development and technology, both on the scale of large

> companies and at individual level, is available through developments that are open to anyone with an Internet connection and a certain knowledge.

This phenomenon can have a major impact on technology transfer between countries that are more or less developed, and internally, between large multinationals and local micro-enterprises.

The possibility of free access to both the code and decisions on design and development offers great potential to small technology companies, which can be in contact with and adopt the most innovative technology backed by large financial investments.

Given their size, these companies generally base their activities on specific niches and require only a few customers to stay in business. The possibilities of market segmentation are endless, but one common factor is that of closer and more personalised attention (many customers prefer to be the big customer of a small business than a small customer of a large multinational).

The more relevant companies of this nature include those that base their differentiation on the use of free software not only for the benefits we have mentioned thus far, but as a **statement of intent**, as yet another element of a business logic that seeks not to accumulate profit but to generate self-sustaining livelihoods through the provision of services that contribute to the development and well-being of society.

The inner workings of these companies also often reflect this philosophy and approach to business, based on **horizontality** and **transparency**. Interestingly, Spain's legal framework provides for a concept of business substantially aligned with what we have described: worker cooperatives, in which there are no financial backers and the workers themselves manage and control the company.

Again, the concept of **business ethics** is neither new nor unique to free software but takes on a special meaning in this type of company. Often, these small businesses form groups through different types of networks, which is a key strategy for encouraging support and cooperation between them, in line with the ethical and political principles on which they are based.

A considerable proportion of the potential customers are other companies with similar operating principles, organisations with social or political motivations, and government bodies.

Examples of this model include several Spanish companies with a similar type of operation, which have been uniting in the Ikusnet group (http://www.grupoikusnet.com/) under the following principles:

> "Our methodology is based on cooperation and 'horizontality' in making and implementing decisions, to the extent that the mode of cooperation itself becomes a productive force that seeks to deliver its effects in the framework of the information and knowledge society."

We can also mention the Madrid-based cooperative **Xsto.info** (http://xsto.info), a micro-enterprise with less than ten workers. Born at the heart of social movements, it was established as a worker cooperative in 2003. This choice of legal form is, like the use of free software, a statement of intent regarding its operating principles, which are complemented by the website, in line with the following motto:

> "There is still time to take part in this social transformation to ensure that it occurs in a participatory, open, free and democratic way".

Among its customers we find local authorities such as Parla City Council, and various types of association, including the Federación Regional de Asociaciones de Vecinos de Madrid (Regional Federation of Neighbourhood Associations of Madrid).

Another very representative example, particularly interesting given its age, is Easter-eggs, which we will now discuss in detail.

### Example 4.9. The Easter-eggs case

| | |
|---|---|
| Company name | Easter-eggs |
| Head office | Paris, France |
| Website | www.easter-eggs.com [http://www.easter-eggs.com] |
| Creation date | 1997 |
| No. of people employed in 2007 | 15 |
| Turnover in 2006 | €800,000 |

**Easter-eggs**is a French SME with a consolidated track record that provides services for free software. Founded in 1997, it offers a wide range of services, from the installation, administration and security of GNU/Linux systems to the adaptation of applications and custom developments and consulting, auditing and training. The company offers services for older free software – and still looks healthy: profitable from the moment it was created, it now employs fifteen people and obtained a turnover of €800 thousand in 2006. Its clients include the René Descartes University of Paris (http://www.univ-paris5.fr/) and Europcar, for which it implemented a GNU/Linux migration programme in 3,500 of its agencies.

For the company, the decision to provide services for free software was based on ethical rather than financial principles, and these principles are also what led it to define a very unique method of business operation. In a manner similar to that of the operation of Spanish worker cooperatives, Easter-eggs is fully and solely controlled by its employees. There are no venture capitalists or foreign investment of any sort. An association was set up to implement this organisational system, the Association of Easter-eggs Employees (http://www.easter-eggs.org), which holds a 99.8% stake in the company.

These were the foundations on which Easter-eggs built its business differentiation, defining itself as a "social company" with a central concern for creating a "citizen-based company" that responds to the growing aspirations of citizens who are beginning to realise the limits of consumerism and demand that companies act with purpose. Its operating principles include financial transparency (its accounting records are available for download from its website: http://www.easter-eggs.org/rubrique_10_Comptabilite.html), equal pay and mechanisms for the involvement and co-responsibility of its employees.

As part of its strategy to create networks and bring together small, socially-responsible businesses to provide services on a larger scale and as a method of joint promotion, in 2002, the Easter-eggs association created the *libre enterprise* network (http://www.libre-entreprise.org), which encompasses approximately sixteen French companies offering free software-based services, all with similar business models.

# Ancillary markets: hardware

One of the first business models described by Hecker, "Widget Frosting" is still as valid today as it was then. For hardware manufacturers, the development of software is a necessary expense if they are to sell their products, so any strategy that will lower the associated costs is desirable. In addition, following a model of free software development extends the possibilities of portability to other platforms, thereby increasing the market segment. We saw earlier how the major providers, which include hardware in their offer, are incorporating free operating systems as a way of reducing the final costs of the service, thus increasing their potential customer base.

On this point, it is interesting to note the role that Linux is playing in the new generation of **embedded devices**. We are witnessing a return to the combined sale of hardware and software in this type of device, which must come with its specific functionality built-in, often with simple operating systems with limited functionality. Nonetheless, the possibility of using embedded Linux has increased the business opportunities for this type of hardware.

## Important

The use of free software offers significant advantages in terms of cost savings, shorter development periods (essential in a market governed by short life cycles), ease of development subcontracting (due to a highly modular existing base) and the possibilities for innovation introduced by setting up a community around the product. Moreover, the use of free software gives manufacturers significant independence from the Windows Mobile and Symbian platforms, and hence, from the agendas of Microsoft and Nokia.

Currently, Linux-based operating systems are the most common in embedded systems and their adoption by consolidated companies of the sector, such as Wind River, points toward the continuation of this trend. In the smartphone market, Linux increased from 3.4% in 2004 to 14.3% in 2005, while embedded Windows only grew from 2.9% to 4.5% in the same period.

### Recommended website

For more information:

*www.os3sl.com/Documents/Seminario_UAM_I.pdf* . Alejandro Lucero, "Seminario UAM: Linux en Sistemas Empotrados". *www.os3sl.com/Documents/Seminario_UAM_I.pdf* .

Furthermore, the existence of software at an affordable price for a large audience also generates an ecosystem of needs around it, which the hardware often forms part of. The **Asterisk** IP voice platform, for example, allows many businesses to use switchboards, with a significant reduction in costs. However, it requires users to purchase certain hardware elements, such as IP terminals, Asterisk cards, routers, recording systems, etc.

The manufacturers of these products can benefit from the spread of software like Asterisk, so they will have much to gain from participating in and contributing to its development. Likewise, software development companies can earn money by selling hardware and related services, as is the case of Digium, the company chiefly responsible for the development of Asterisk.

There are also other spaces and niches that can be exploited through this technology, such as those tapped by **Avanzada7**. This Málaga-based company sells the necessary hardware for the implementation of Asterisk, but acknowledges that it is neither a manufacturer nor a major distributor. Its differentiation stems from the provision of free support services following the sale of the devices. Avanzada7 has also established a partnership with Digium, the company responsible for the development of the software, creating a trusted network that extends to other companies wishing to implement Asterisk for end customers. Thus, it has set up a pyramidal network of the type described above based on the needs generated by free software, which it exploits through *coopetition* strategies.

### Example 4.10. The Chumby case

| | |
|---|---|
| **Company name** | Chumby Industries, Inc. |
| **Head office** | San Diego, CA<br><br>(United States) |
| **Website** | www.chumby.com [http://www.chumby.com] |
| **Creation date** | 2005 |
| **No. of people employed in 2007** | |
| **Turnover in 2006** | |

**Chumby Industries** was set up with the aim of creating and marketing the "Chumby", launched in August 2006. This wireless (Wi-Fi) device was designed to replace the clock radio and can connect to the "Chumby Network", where it can download different types of information. It can play podcasts, Internet radio, and some videos. The device runs Linux and Flash Lite, an Adobe program with small interactive applications or "widgets". It does not have a browser and contents can only be downloaded through widgets, each of which has its own specific function: read the latest news from a blog, download the latest photos from a gallery, etc.

The Chumby hardware and software are free and both its schemas and printed circuit boards – and even its source code – can be downloaded. The company's marketing activity is based on its openness: the Chumby can be customised at any level by changing the outer casing and (literally) sewing on extensions to taste, creating new widgets or hacking the hardware. Thus, the device is not only sold as "user-friendly", it also opens the door to the expansion of its features beyond the control and financing of the company, leaving it to evolve into what every user wants it to be.

Nonetheless, Chumby's business model is not aimed at obtaining revenue from hardware, and the price of the device is relatively low. Steve Tomlin, founder and CEO of the company argues that several business models were possible with Chumby: they could have charged more and followed the model of a traditional hardware vendor, with the problems of recurring revenue that this would generate, or they could charge little for the device, but then charge for content subscriptions. However, the company preferred a third way: to obtain the revenue needed to just cover costs with sales and generate its profits through advertising.

To secure this new field of business, Chumby is not 100% open and there are constraints on its use, both in the hardware and on the "Chumby Network", thus guaranteeing the business model.

**"Chumby network" access**

After purchasing a Chumby, the user must register on the company website to access the widgets, accepting their terms of use. These terms allow anybody to add new widgets with any type of information they wish, giving their permission to distribute this information to any device connected to the network. However, restrictions are placed on permitted content, and inappropriate content (racist, violent, sexist, spam, etc.) is banned, as is commercial content:

"**Prohibited Content** includes Content that: (...) except as expressly approved by Chumby, involves commercial activities and/or promotions such as, without limitation, contests, sweepstakes, barter, advertising, or pyramid schemes." (http://www.chumby.com/pages/terms)

A payment must therefore be made to obtain authorisation for advertising content. The terms and conditions also warn that the user will receive advertising when he/she connects to the Chumby network.

Although widgets can technically be incorporated outside the Chumby network using USB devices, the company is confident that most of the contributions will remain within its network, thus attracting enough content to generate value from the number of people and contributions on it.

**The device**

Chumby allows access to the schemas and PCBs of its device. However, manufacturers seeking to use its designs and incorporate them into their own products have to pay the company to licence their new product. In addition, they have to accept that, besides any other networks to which they connect, they will also incorporate the Chumby Network.

To summarise, Chumby acknowledges that the value of its device lies in the content, in a manner similar way to O'Reilly in "Open Source Paradigm Shift" and others. Its strategy, besides characterising the product by its openness, is to attract as many people as possible to the network in an attempt to make it a benchmark network for small mobile devices of this nature. However, instead of selling content through subscriptions, it has decided to capitalise on this value through advertising.

For the company, the use of open hardware and software is a key strategy for the spread and adoption not just of its device but of the network that it has created to provide content. Moreover, its openness gives it a clear differentiation and commercial edge over similar products like Apple's iPod Touch and iPhone.

## Other ancillary markets

The increased spread of free software, both due to its form of development and its use, generates other related markets that have been exploited by diverse companies:

- **Community and development:** perhaps the most obvious examples are those that provide hosting services and collaborative tools for software projects, such as SourceForge, CollabNet or Freshmeat. There has also been a proliferation of code search engines, including Google Code, Koders, Krugle and Codase.

- **Legal certification:** companies offering this type of certification are also becoming increasingly relevant. They ensure that a software or particular combination is legally possible and are familiar with the licensing problems it could have. This service is provided by the companies we saw earlier, such as the creators of platforms and bundles, like SpikeSource, but others have sprung up that focus entirely on legal issues, such as Black Duck and Palamida.

- **Sale of books:** O'Reilly and his books are one of the most often cited examples in this category.

- **Merchandising:** we should not overlook the importance of merchandising as a supplementary or even main form of financing. Examples include ThinkGeek, a subsidiary of SourceForge, which contributes revenue through Internet sales of various types of product for "geeks": from t-shirts and mugs to a range of gadgets.

# abstract

This module has looked in detail at the diverse valid and viable business models based on free software. The growth of companies that focus entirely on its exploitation and the redirection of the strategy of software multinationals is conclusive proof.

Initially, we described different classifications proposed by a range of authors over time:

- The classifications of Hecker and Raymond, based on the observation of companies that used free software as part of their business models.

- The classification of the European Working Group on Libre Software, based on the business financing model.

- Daffara's classification, based on empirical studies.

Finally, we proposed and developed our own business models proposal:

- Specialist/vertical, focusing primarily on the free software product and which can adopt mixed dual licensing models, proprietary accessories, distributed product sales or service provision models for the product, such as software as a service.

- Associated services such as custom developments, product selection, installation, integration, technical certification, training, support and maintenance, which may be oriented towards the distribution of platforms, large scale integration or the service of small businesses and micro-enterprises.

- Ancillary hardware markets, which use free software to complement their main business: the sale of physical products or the business of contents accessible from a particular hardware.

- Other ancillary markets, such as collaborative tools, legal certification, the sale of books or merchandising.

# Bibliography

## bibliography

Augustin, L.. 2007. "A New Breed of P&L: the Open Source Business Financial Model". Open Source Business Conference (OSBC)*Metcalfe, Randy*. . < http://www.osbc.com/live/images/13/presentation_dwn/A_New_Breed_of_P_and_L.pdf > [Consulted in February 2009]

Mickos, M.. 2007. "Open Source: why freedom makes a better business model".*Open Source Business Conference (OSBC)*. . < http://www.osbc.com/live/images/13/presentation_dwn/Keynote-Open_Source_Why_Freedom.pdf > [Consulted in June 2008]

West, J. and Gallagher, S.. 2006. "Patterns of Open Innovation in Open Source Software". In: Henry Chesbrough; Wim Vanhaverbeke; Joel West (eds.).*Open Innovation: Researching a New Paradigm*. (pp. 82-106). Oxford: Oxford University Press. < http://www.openinnovation.net/Book/NewParadigm/Chapters/index.html > [Consulted in June 2008].

Capiobanco, F.; Onetti, A.. (July 2005). "Open Source and Business Model Innovation. The Funambol case". In: M. Scotto; G. Succi (eds.).*Proceedings of First International Conference on Open Source*. (OSS2005) (pp. 224-227). Genoa. < http://oss2005.case.unibz.it/Papers/4.pdf > [Consulted in June 2008].

Riehle, D.. 2007. "The Economic Motivation of Open Source Software: Stakeholder perspectives".*IEEE Computer*. (vol. 4, no. 40, pp. 25-32). < http://www.riehle.org/computer-science/research/2007/computer-2007-article.html > [Consulted in February 2009]

Comino, S.; Manetti, F. M.. 2007. *Dual licensing in open source markets*. . Università Degli Studi di Trento, Department of Economics. < http://www-econo.economia.unitn.it/new/pubblicazioni/papers/18_07_comino.pdf > [Consulted in June 2008].

Daffara, C.. 2007. *Business models in FLOSS-based companies*. . Conecta Research, 2007. < http://opensource.mit.edu/papers/OSSEMP07-daffara.pdf > [Consulted in June 2008]

**Pal, M.** (July 2005). "Participatory Testing: The SpikeSource Approach". O'Reilly Network. <http://www.oreillynet.com/pub/a/network/2005/04/07/spikesource.html> [Consulted in June 2008]

Kelsey, J.; Schneier, B.. (June 1999). "The Street Performer Protocol and Digital Copyrights".*First Monday*. (vol. 4, no. 6). < http://www.firstmonday.dk/issues/issue4_6/kelsey/ > [Consulted in June 2008]

Rasch, C.. (June 2001). "The Wall Street Performer Protocol".*First Monday*. (vol. 6, no. 6). < http://www.firstmonday.org/issues/issue6_6/rasch/index.html > [Consulted in June 2008]

Daffara, C.; Barahona, J. B. et al. 2000. "Free Software/Open Source: Information Society Opportunities for Europe?"*Working paper*. . < http://eu.conecta.it/paper/ > [Consulted in February 2009]

**Lucero, Alejandro.** "Seminario UAM: Linux en Sistemas Empotrados". <www.os3sl.com/Documents/Seminario_UAM_I.pdf [http://www.os3sl.com/Documents/Seminario_UAM_I.pdf]> [Consulted in June 2008]

Raymond, E.. 1999. *The Magic Cauldron*. < http://catb.org/~esr/writings/magic-cauldron/ >

Spanish translation in: <http://gnuwin.epfl.ch/articles/es/magiccauldron/es-magic-cauldron/es-magic-cauldron.html> [Consulted in February 2009]

Hecker, F.. 1998. *Setting Up Shop. The Business of Open Source Business*. < http://hecker.org/writings/setting-up-shop > [Consulted in February 2009]

Metcalfe, Randy. 2006. *Open Source Business: Differentiation and Success*. < http://www.oss-watch.ac.uk/resources/businessmodels.xml > [Consulted in February 2009]

**Case studies**

*50 Open Source Success Stories in Business, Education, and Government*. <http://www.blogcrm.com/50-open-source-success-stories-in-business-education-and-government.php>

Red Hat and J. Boss. "Is Open Source viable in Industry? The case of Red Hat and JBoss". <http://www.whyfloss.com/es/conference/madrid08/getpdf/68>

Avanzada7. "Business models based on Asterisk: The case of Avanzada7". <http://www.whyfloss.com/es/conference/madrid08/getpdf/64>

Openbravo. "Openbravo: keys to success in free software application development". <http://www.whyfloss.com/es/conference/madrid08/getpdf/49>

Liferay. "Liferay Enterprise Portal: The project, the product, the community and how to extend it". <http://www.whyfloss.com/es/conference/madrid08/getpdf/66>

Various cases. <http://www.opensourceacademy.gov.uk/solutions/casestudies>

# Chapter 5. Developing free software in companies

Amadeu Albós Raya

GNUFDL
2010-02-01

# preface

In this module, we delve into the world of free software production and its most relevant features for the product, company and user community.

To begin with, we discuss the development of free software from the point of view of the project, considering the main aspects affecting the population and management of the project, and the participation of the user community in a variety of aspects.

The free software project formalises the relationship between the company and the user community. The adaptation of the specific features of this relationship is essential to achieving the aims of the project.

We then move on to describe the specific features of the free software user community and its management by the company. This management complements the production methodology and implements the relational strategy discussed earlier.

Finally, the module concludes with a case study of a real company that produces free software.

This module is structured as a guide for external reading, the aim of which is to provide more detail on the features of the various aspects that emerge and which are relevant to free software business production.

# objectives

After completing this module, students should have achieved the following aims:

1. To be familiar with the methodology of free software production.
2. To understand the importance of the user community for the development of products based on free software.
3. To identify and analyse the relevant factors affecting the success of free software production.
4. To understand the importance of formalising a methodology to complement the efforts of the company and the user community.
5. To obtain a deeper understanding of the direct and indirect implications of carrying out a development project based on free software.

# Free software production

In this first section, we will focus on the production of free software from the perspective of its development or creation, i.e. without considering the possible business models that exploit it for profit.

Several subjects of this Master's degree, particularly those on software production, discuss the technological process characterising free software development at length.

### Important

This technological process supplements the methodologies allowing us to formalise a viable cooperative project that will last over time. In this sense, the cooperation of the user community on the free software project is crucial for obtaining a critical mass of users to enable the project to be viable.

Consequently, many of these methodologies and actions are designed to offer support and guarantees to relations between the project and the user community. To understand the importance of this relationship, we can simply visit the resources offered by the more popular free software projects to the user community.

#### Example 5.1. Popular projects

For example, OpenOffice.org (http://contributing.openoffice.org/) and Mozilla (http://www.mozilla.org/contribute/).

To develop these concepts, over the next few sections we will describe three complementary points of view. First of all, we will consider some basic ideas on free software production. We will then briefly detail the main steps to take to implement a project based on free software. Lastly, we will detail the main aspects of free software project management.

# Free software production

### Important

The production of free software, like the production of any software, responds to the need to solve a specific technology problem.

Although the technological process of refining and developing a free software application may share many similarities with an application based on proprietary software, the difference marked by the openness of the model gives it a special type of operation. In other words, the open and cooperative nature of its production affects the structure of quantitative and qualitative evolution down the versions.

Many authors have written about the specifics of producing free software. Since it is not the aim of this module to detail or describe these features at length, given that they are comprehensively dealt with in other subjects, we will focus here on pointing out some of the more interesting ones in our case.

To do so, we will consider some of the concepts in Eric S. Raymond's paper *The Cathedral and the Bazaar*, which analyses the special features of free software, particularly GNU/Linux.

#### Recommended website

E. Raymond. 1997. *The cathedral and the bazaar*. ( http://www.catb.org/~esr/writings/cathedral-bazaar/ ).

- **The origin of production**

  Broadly speaking, the production of free software emerges from the particular needs of the user or developer in his or her daily activity. In other words, collaboration on the development of the software begins when we look for and find a problem that we want or need to resolve.

### Example 5.2. Early stages of production

The bulk of the foundations of free software are based on the publication of specific adaptations or developments made by workers in the performance of their daily work.

- **The user community**

  The free software user community, which includes both end users and developers and programmers, is the pillar that gives meaning to the definition of free software development.

  Treating users as partners in the production project is the easiest way to debug and improve the code quickly (if the collaborator base is big enough).

  Thus, collaborators are one of the most valuable resources for the development of the application, so it is also helpful to recognise good ideas and the solutions they provide.

- **Versions of the application**

  One of the features of free software production is the reuse and rewriting of the original code to create a new code that is either error-free or which has new features or improved performance (among other aspects).

  Moreover, free software development projects encourage the quick and regular release of the code, which means that the project activity is dynamic and continuous.

- **Coordination of production**

  The individual – or individuals – who coordinate/s the project must be able to manage the global potential of the user community, guiding the project's evolution without coercion and taking advantage of the resources and synergies offered by networks such as the Internet.

  The legacy of the application's code and coordination management are important for the future of the free software development project. The choice of a successor to control and manage production should not be left to chance.

# The free software project

## Important

In addition to the technological and functional considerations of applications based on free software, one of the primary aims of any free software project is to disseminate the application or obtain a critical mass of users.

To put it another way, it is not very helpful for the future of the project if the generated code is not known and applied by potential users, even if specific problems or shortcomings have been addressed. This is also a necessary aim for its subsequent maintenance and evolution over time. In the case of free software, fulfilment of this aspect is essential for the creation of a stable and lasting user community.

Several guides have been written that, to a greater or lesser extent, contribute the necessary concepts for the creation and management of projects based on free software. In this section, we will develop this issue using Benjamin Mako's *Free Software Project Management HOW TO* article, which reviews the main features of the project from a practical angle.

### Recommended website

B. Mako. 2001. *Free Software Project Management HOW TO*. ( http://mako.cc/projects/howto ).

**Launch**

Before launching a project based on free software, it is very important to design a solid structure that will withstand the subsequent development process with sufficient guarantees.

In general, the basic structure of the project must take into account the following:

- The need to create a new project, either with its own ideas and aims or through existing, related projects.
- The definition of the main features of the application (functionality, licensing, numbering, etc).
- The basic infrastructure to support dissemination of the new project and collaboration on its development (website, contact e-mail, etc).

**Developers**

Once the project has been launched, the next aim will be the integration and consolidation of the users and developers of the application. We must create policies and strategies allowing us to define and structure the collaboration of the latter.

Cooperation policies must meet two main aims:

- The coordination of internal and external production, including the delegation of responsibilities and protocols of acceptance for contributions.
- Production management, such as the structure of development branches and their associated repositories.

**Users**

With products based on free software, the users are often developers too (and vice versa). One of the main aims to take into account then are application tests, be they functional, operational, quality, etc.

**Support infrastructure**

The daily activity of a project based on free software could not be carried out without a support infrastructure adapted to its cooperative aims.

The key actions in this regard are carried out during the project launch. However, once it is up and running, we will need to adapt, improve and supplement the existing resources in line with the progress of the project.

## Example 5.3. Usual resources

Some of the most common resources in free software projects are: documentation, mailing lists, bug tracking systems and versions, forums, chats, wikis, etc.

**The application**

Undoubtedly the most important component of the project is the application, on which the rest of the aspects considered are based. One of the key features required by an application is for the user to have sufficient guarantees on the performance of each version released.

The release of versions is a sensitive issue that requires careful thought. Broadly speaking, we need to consider the following:

- Control of revisions for functionality and debugging (alpha and beta versions, candidate distribution, etc).

- When to launch the full version, i.e. when the code will be ready to offer guarantees that we and the users expect.
- How to release the version (packaged, source code, binary, etc).

**Dissemination of the project**

Lastly, as we initially explained, raising awareness of the project is important, but this task should be carried out taking into account whether we will want to reinforce its foundations over time.

As the project progresses, we need to think about publicising it in free software mailing lists or on *Usenet*, including the project in other public portals (such as *Freshmeat* or *SourceForge*), or advertising new versions of the application in the project's own mailing lists.

# Project management

In this section, we will describe in detail the aspects of project management that, as founders of the same, we must keep in mind to guarantee success.

The concepts we describe in this section supplement those of the above sections, since they allow us to specify and improve the various actions considered. Hence, it is possible to find direct and indirect coincidences with these arguments.

To indicate the basics of the management of projects based on free software, we will take into account the considerations set down in Karl Fogel's *Producing Open Source Software*, particularly Chapter 5, entitled "Money".

## Recommended website

K. Fogel. 2005. *Producing Open Source Software: How to Run a Successful Free Software Project* . (Chapter 5 "Money"). ( http://producingoss.com/en/money.html ).

**Funding**

The special features of free software projects mean that many contributions are informally subsidised (for example, when a company employee publishes the adaptations it has made to the code during his/her daily activities).

Donations and grants are also made, contributing direct income to keep the project going, but we must take into account the management of these funds, since much of the support afforded to a free software project is based on the credibility of its participants.

**Types of participation**

There are many types – and possible combinations – of financial participation in a free software project. This funding model also influences aspects that depend not only on the project but also on its environment and context of action.

Broadly speaking, participation in a free software project is related to the collaboration of its participants, the business model exploited by the company that promotes it (where applicable), the marketing activities undertaken, the licensing of the products involved and the donations made.

**Open-ended contracts**

The application's team of developers is very important for the development of the project and its future evolution. The stability and permanence of the participants in their posts of responsibility will strengthen the foundations and credibility of the project vis-à-vis the user community.

## A stable project

Credibility is essential for all actors directly or indirectly involved in the project, since this cannot be transferred to substitutes. Moreover, loss of credibility can affect the future of the application and the project to varying degrees, so we need to take the appropriate measures to actively monitor and manage the project.

**Decentralisation**

One of the most relevant – and desirable – features of free software user communities is the distribution and decentralisation of the decisions taken in the project.

Hence, the project organisation should consider this structure as a way to motivate and strengthen the community of application users, ensuring that the consensus emerges from interaction between its members.

**Transparency**

The above aspect of decentralisation gives us an idea as to the transparency and justification that should exist in the relationship between the project and the community.

Both the aims of the project and lines of evolution of the application must be clear and well known to all those involved in it. The influence of the founder on future behaviour must be exercised in a sincere and transparent way in order to guarantee the credibility of the project.

**Credibility**

Project credibility (both overall and of its individual members) has cropped up in a number of the issues we have already discussed. Its relevance is closely related to the free software user community and it is an important prerequisite for maintenance of the project over time.

Money or a hierarchical position cannot generate the necessary credibility in the actions of individual members at any given time. In other words, the established methodology, procedures or protocols, or the workings or operation must be the same for everybody, without exception.

**Contracts**

Employee hiring is another aspect to take into account, particularly in free software projects, due to its impact on structure and operation. We need to ensure that all of the details and processes of recruitment are open and transparent.

In fact, it is important to review and approve these changes with the collaboration of the user community, to the extent that, in some cases, it may be preferable or desirable to contract developers directly from the community with write permissions on the official repository (committers).

**Resources**

Free software projects are based not only on the evolution and maintenance of the code of an application based on free software; they must also consider additional aspects of support.

### Example 5.4. Additional resources

This is the case of the quality management of the code produced, the legal protection of contributions, the documentation and utility of the application, and the provision of infrastructure resources for the free software community (websites, version control systems, etc).

These resources can generate significant differences in the dissemination and popularisation both of the application and of the project in the free software user community.

**Marketing**

Lastly, although we are dealing with a project based on free software, we should implement marketing measures for the dissemination and popularisation of the application and of the project as a whole.

Hence, we must remember that the full workings of the project are in the public eye and that each of the claims made may be easily demonstrated or proved wrong. The establishment of measures to control the image and operation of the project must enable it to gain credibility, transparency and verifiability.

These measures include the importance of maintaining an open, honest and objective policy on rival projects. Firstly, because it encourages a certain value for the user community, and secondly, because it fosters the development of coopetition strategies with aligned projects.

# The user community

As explained in the first section of this module, the role of the free software user community is very important in the paradigm of free software development.

### Important

Both the users and the developers who form part of the community collaborate on the maintenance, support and evolution of the application over time, thereby ensuring the cohesion and stability of the project.

Consequently, their participation is essential for securing the project aims and should be considered as such by any money-making organisation seeking to exploit a business opportunity based on the production of free software.

In this sense, the relationship between community and business should be founded on the credibility and transparency of all actions and decisions taken, so that both parties can benefit from the relationship. Not surprisingly, the company's positioning with respect to products based on free software must be well defined and structured to encourage the creation of a community of collaborators around it.

Note that the user community is a dynamic organisation that evolves over time, so it will be necessary to set up management methodologies in order to maintain an optimal relationship. This management includes the establishment of procedures to identify the current status of the community, to assess the quality of contributions to the project by members, and to define legal aspects affecting these contributions.

The following sections will study each of these aspects in turn.

# Community management

### Important

To secure the aims of the project, a company that undertakes a free software development project must organise its relationship with the user community carefully.

In the first section of this module, we looked at the main aspects underpinning a free software project. If a company acts as project founder, it will need to establish and organise a strategy to suit the business aims, though bearing in mind that it has to compensate for the collaboration it hopes to obtain from the user community.

Hence, as with any other free software project, issues such as credibility and transparency, among others, have a very important role in creating a community of users around the project.

Ben Collins-Sussman and Brian Fitzpatrick have identified and classified the different *Open Source* strategies that can be adopted by a company based on free software development at the OSCON 2007 conference entitled "What's in it for me?".

### Recommended website

**B. Collins-Sussman; B. Fitzpatrick** (2007). "What's in it for me?"

(http://www.youtube.com/watch?v=ZtYJoatnHb8).

This classification characterises the two main components of the relationship between company and community:

- On the one hand, the orientation, structure and general operation of the project, and the company's responsibility in this.
- And on the other, the benefits and drawbacks for the company and the user community resulting from the selection of a specific strategy to implement the project.

Hence, the work of Collins-Sussman and Fitzpatrick is a guide to best practices in formalising a healthy relationship between the company and the user community.

In the following sections, we will briefly introduce the main features of this *Open Source* strategy classification.

#### Fake Open Source

## Important

This strategy is based on opening or releasing the application's source code under a licence not approved by OSI.

### Recommended website

Open Source Initiative (http://www.opensource.org/).

It is not really an *Open Source* strategy because not only are thebenefits lost, but some members of the community may even boycott the project.

Nonetheless, the project can obtain media coverage and attract attention with a relatively low effort and cost.

#### Throw code over the wall

## Important

This is a similar strategy to the one above except that this time the company opens or releases the code under an OSI-approved licence, although it is still not concerned or does not accept responsibility for the future of the project.

In other words, by opening up the code and forgetting about it, the company portrays an image of poor credibility, since it releases an application for which there is no user community to keep the project alive. In this case, alternative communities may spring up to develop the software outside the business goals.

**Develop internally, post externally**

### Important

This strategy is based on developing the application internally within the company and publishing the progress in a public repository.

This time, the company improves both its public relations with the user community and its credibility in the world of free software. For its part, the community could collaborate on the project from time to time. Nonetheless, a totally internal development will encourage the development of parallel communities that do not follow the business calendar (which generates an element of distrust).

**Open monarchy**

### Important

This strategy is based on making public both discussions and the repository of the application, although the users with the rights to it are from the company.

In this case, the credibility and transparency of the companies and the input from the community are improved (which results in better code) but the company still has the final say on all decisions made. This constitutes a risk to the long-term maintenance of the community, including the possibility of a fork in the project.

**Consensus-based development**

### Important

This strategy exploits almost all possible relations between company and community, given that virtually everything is done in public.

In this case, the project is based both on distributed and decentralised decision-making and on meritocratic work systems among collaborators.

These features produce a model with high quality volunteers that is sustainable in the long run, since the company gains in credibility, transparency and reliability in the eyes of the community and other free-software companies.

Nonetheless, the short-term benefits are limited and the workload is significant. In this case, the role of the project leaders is relevant for the strategic operation of the entire organisation.

# Community features

The community of free software users is a dynamic and evolutionary organisation in the sense that there are several factors that influence and shape its situation and future trends to varying degrees.

### Important

When considering a free software project, it is desirable to create an early and strong user community around the application, given that part of the success and aims of the project depend on it.

Once the community has been created, it is important to schedule activities that will not only keep it stable but also enlarge and evolve it, at least at the same pace as the product. Before we take any action in this regard, we need to ascertain the current status of the user community and its recent evolutionary trend in relation to the project.

Accurately identifying the current status of a user community can be relatively complicated in practice, mainly due to its qualities of distribution and decentralisation.

Nonetheless, we can take into account a series of indicators that will allow us to establish a sufficiently realistic approach for making decisions on this subject.

The article "Assessing the Health of a FLOSS Community," by Crowston and Howison, describes a simple but effective guide to identifying and assessing the status of a community of free software users. This guide considers the main indicators that should be taken into account when assessing the health of the community and, by extension, the free software project.

### Recommended website

**K. Crowston; J. Howison** (2006). "Assessing the health of a FLOSS Community" (http://floss.syr.edu/publications/Crowston2006_Assessing_the_health_of_open_source_communities.pdf [http://floss.syr.edu/publications/Crowston2006Assessing_the_health_of_open_source_communities.pdf])

The following sections will briefly introduce some of their findings.

**Life cycle and motivations**

Diverse authors concur that projects are initiated by a small group of founders before being structured and publicly developed.

Once the project has been launched, a second phase should begin allowing for the progressive refinement of the initial concept. In other words, the sharing of ideas, suggestions and knowledge must revolutionise the original concept. This process cannot be completed without the cooperation of the free software community.

Moreover, the participation of members of the community in the project is chiefly motivated by intellectual development, the sharing of knowledge, interest in the application, the ideology or philosophy behind the project or free software, reputation and community obligation.

**Structure and size of the community**

The user community of an application based on free software can be structured in many ways, taking into account the actions and decisions of the project founders and the features of the application and/or its production.

In general, we can consider an application's user community to be healthy if it has a functional hierarchical structure aligned with its aims around an active core of developers.

## Hierarchical structure

This concept can be compared to the structure of the layers of an onion (onion-shaped), whereby the most active members of the project are at the core and the less participatory members are found in the outermost layer.

Broadly speaking, we can identify the following types of member in a project:

- Developers of the application kernel, with write permissions on the repository and a significant history of contributions to the project.
- Leaders of the project, who motivate and lead the project and its user community to maturity and stability.

- Developers in general, who contribute code but have no write permissions on the repository. They often perform review tasks.
- Active users, who test the application, report bugs, draft documentation and link the project up with passive users, among other activities.

### Note

This initial classification of typologies is not a closed structure, since each project will adapt it to suit its particular features.

**Development processes**

The process of free software development can often be inadequately formalised in projects, mainly due to the absence of roadmaps, explicit work assignment or the lack of prioritisation in the application's features.

The organisation of the project is relevant to the functioning and coordination of production, although a certain degree of duplication of effort could be considered a positive sign of the relationship and involvement of the community with the project.

Likewise, the cycle of evaluation and subsequent acceptance of contributions from community members to the project provides accurate information on its health. For example, the rejection of a contribution can reveal a cohesive and qualitative vision of the project in the long run.

# Quality management

The quality of free software has sometimes sparked debate between its advocates and detractors, particularly concerning aspects such as the openness of the development model or the skills level of collaborators who contribute to the project, for example.

### Important

As with any software project, free software production should establish measures for quality control throughout its life cycle. In other words, we must be able to assess its quality and compare it with the levels expected at any stage of production or exploitation and from any angle (founders, users or community).

While the openness and decentralisation of the model of free software development allows for quality control and management mechanisms, they are not a solution in themselves and planning should not be overlooked because of these features.

To develop the quality aspects of free software production, we will study the Dhruv Mohindra's article "Managing Quality in Open Source Software", which conducts a detailed analysis of quality control in free software environments. In the subsequent sections, we will review the main ideas of the article.

### Recommended website

**D. Mohindra** (2008). "Managing Quality in Open Source Software"

(http://www1.webng.com/ dhruv/material/managing _quality_in_oo.pdf [http://www1.webng.com/dhruv/material/managing_quality_in_oo.pdf]).

**Quality in free software**

In general, the quality of a software solution can be assessed both by its architecture or internal design and by the functionality it provides to the user.

The specific features of openness and decentralisation of the free software development model create an infrastructure that allows for quality management policies to be established through the identification and resolution of problems, among other aspects. Still, a lack of clarity and/or structure in production processes can sometimes generate unexpected results.

**Assessing quality**

There are several formal methodologies and metrics for assessing the functional quality of an application. Quantifiable metrics depend largely on the typology of the software itself, so they must be chosen in accordance with the features and aims of the application.

The free software community plays an important role in non-quantifiable quality: firstly, in the tests performed by the quality team, and secondly, in the activity of the users of the application, who report evidence of malfunctions or for product enhancement.

In this sense, the decentralised and distributed nature and operation of the user community is important for increasing the quality guarantees of the production process.

**Control and review**

An important factor in end product quality is the control and review of the entire development process. In general, free software production projects use version control systems to efficiently and effectively support the evolution of the diverse project components.

There are different ways to organise the control and review of the evolution of the software and its branches of development and repositories, among other aspects. In all events, though, it is a good idea to adapt the production methodology and systems for the control and review of changes to the specific features of the project and the product being created.

**Free software myths**

Despite the passing of time, there are still some myths, both positive and negative, associated with free software that can influence its assessment to different degrees.

These myths have no solid foundation on which to base a coherent and sustainable quality management, so we need to identify and evaluate each one individually.

We will then discuss some common myths associated with the quality of free software.

- The fact that the source code is public does not guarantee that it is secure and/or of good quality, as this depends on the community interest and reviews.
- Feature freezing does not increase the stability of the application in itself, because the important thing is to write good code from the start.
- The best way to understand a project is not to correct its possible shortcomings, as the documentation is significantly better for this purpose.
- Generally, users do not have the latest version of the repository with updated bug-fixing.

Broadly speaking, the testing and review processes, and the public discussions and hacker culture specific to the user community must be complemented by the active planning and management of production quality.

This management should seek to fill any gaps in one or more aspects of the product, e.g. production planning, development of the features or the documentation of the application.

**Additional quality considerations**

In general, both the release of the source code and the incorporation of error handling systems and the sharing of responsibility for the product among all those involved are key aspects of quality management.

Hence, it is also important for the overall quality of the project to consider transparency in all actions, trust the development team, review and test all parts of the source code and promote both the peer-to-peer philosophy and the importance of doing things well from the start.

# Legality and contributions

In a project based on free software with participation from the user community, the legal management of the contributions of each member involved is particularly important.

## Important

This management is crucial both for the project founders and for the members of the community, as it establishes the features of the authorship and ownership of the rights to the resulting code. Its relevance is also strongly influenced by the implications that the combination of codes from different authors could have on a single product.

To develop these concepts, we will refer to section 2.4 "Authors and holders of rights" of the teaching materials for the subject *Legal aspects and the features of exploitation of free software*.

### Recommended website

M. Bain et al.. 2007. *Aspectos legales y de explotación del software libre*. . Universitat Oberta de Catalunya

(http://ocw.uoc.edu/informatica-tecnologia-i-multimedia/aspectes-legals-i-dexplotacio/materials/).

**The author**

## Important

The author of a work is the natural or legal entity that creates the work, so authorship of the original creation is irrevocably assigned to this person.

With works by several individuals, there are a number of possible situations:

- A collaborative work is the unit result of a composition of different parts that can be exploited independently.
- A collective work is the collection of diverse contributions that cannot be exploited independently.
- With a commissioned work (or one with financial compensation), authorship lies with the person or entity that carries out the commission.

In free software, authorship depends largely on the above considerations, taking into account that the transfer of ownership can sometimes be useful and practical.

Moreover, the conditions under which derivative works are created (pre-existing content) may vary materially because of both the author and the work itself. In all events, free licences must specify the conditions of the derivation and redistribution of the works.

**The original owner and the derivative owner**

The original owner of the work is always the author. However, some rights over the work may be transferred to other individuals or entities.

In this case, the recipient of the transfer of part of the rights to a work becomes the derivative holder thereof. Note that only the holder of a particular right may licence that right.

**Identifying the holder**

In order to exercise the above rights, we must be able to identify the author of each work. This can be difficult in free software because the contributors to the project may be many and varied.

To solve these problems, projects based on free software keep lists of the authors who have contributed to them. Sometimes, these projects may require the transfer of all or part of the rights before the contribution can be accepted.

# Case study

In the previous sections, we have examined both free software projects and the management of user communities. Both sections describe the key aspects of free software production from the point of view of project management.

To complete the module, this section will go into further detail on many of the ideas and proposals described above before moving on to study a specific case of a company based on free software.

The following sections are intended to serve as a guide for identifying and clarifying how a company based on free software production implements its particular methodology, formalises and manages its relationship with the user community and addresses the many decisions that need to be taken as time goes on.

In this section, we will study the case of Openbravo, S.L.

# The company

Openbravo, S.L. is a company that develops professional solutions based on free software for business.

## Note

The information in this section has been taken mainly from the corporate website (http://www.openbravo.com/).

**Business model**

The business model exploited by the company is that of providing services for the products it develops. As we explained in other modules, its business strategy is based on associationism and coopetition between companies in order to exploit the same business opportunity.

**Business strategy**

The business model is implemented by partners that provide services to end customers (such as customisation and support). In a sense, this particular hierarchy between producer, distributor (or partner) and client establishes an atmosphere of cooperativism with common goals.

To complete the strategy, the company publishes a manifesto as a sort of statement of intent, which combines aspects of free software (for example, transparency, openness and collaboration) with the company's third-party commitments (such as free access or contribution management).

Management and leadership The company's management combines tasks that are internal and external to the organisation, both in its management team and its Board of Directors, which is the result of foreign investment injected into the company combined with its particular methodology based on free software.

# Products

Openbravo produces two free software solutions that can work independently of each other or in combination. Both products are distributed under free licences and can be downloaded directly from the Internet.

The products offered by Openbravo are:

- **Openbravo ERP**

  Openbravo ERP is an enterprise resource planning system in a web environment that integrates various management functions, such as supply, warehousing, production and accounting, in a modular way.

  The product is licensed under MPL 1.1 and can operate in different environments and database systems and be integrated with Openbravo POS.

### Recommended website

Mozilla Public License 1.1 (http://www.mozilla.org/MPL/MPL-1.1.html).

## Main features of Openbravo ERP

http://sourceforge.net/projects/openbravo/.

One of the highlights of the vast amount of information provided on the product is the roadmap of the project development.

- **Openbravo POS**

  Openbravo POS is a point-of-sale terminal system that can be integrated with Openbravo ERP.

  The product is licensed under the GNU/GPL licence and can run in different environments and with different database systems. It was especially designed for touch-screen terminals.

### Recommended website

GNU General Public License (http://www.gnu.org/licenses/gpl.html).

## Main features of Openbravo POS

http://sourceforge.net/projects/openbravopos/.

The available product information includes the roadmap of the project development.

# The user community

The community of free software users plays an important role in Openbravo's business strategy. The following sections examine its main aspects.

**Open Source Strategy**

To identify Openbravo's Open Source strategy, we need to consider the specific features of the product development methodology and the business structure used to exploit them.

The kernel of both products is primarily developed internally within the company and public repositories and an active user community are maintained around it. For the development of complements, customisations and extensions to the original product, both the user community and partners play a part.

Partners must be examined separately because they correspond to the exploitation of an opportunity by a different organisation.

Nonetheless, Openbravo uses an Open Source strategy that combines different orientations:

- In its product development and review, the strategy used is similar to *Open Monarchy*, mainly due to the internal development of the product kernel, the public repositories of source code, the company's final acceptance of changes to the kernel and the planning of product development (for example, the established roadmaps).
- In the development of complements (documentation, etc.), the strategy is more similar to *Consensus-based Development*, due mainly to its development within the user community.
- And lastly, the strategy for the development of extensions and customisations depends on the developer who implements them. If they are projects carried out within the community (using the resources offered by Openbravo), they are possibly more similar to the *Consensus-based development* model, while if they are developed by partners, they will depend both on the strategy in question and the features of the development.

## Example 5.5. Partner strategy

If the partner develops extensions of the original product, the *Open Source* strategy will depend as much on its business philosophy as on the features of the product (for example, the MPL is more flexible with proprietary modules than the GPL).

**Community structure**

The Openbravo ERP user community is defined and structured as a meritocratic system: there are several levels of collaboration and each is defined by the knowledge required for this level, the amount of contributions made, responsibilities and privileges.

For Openbravo ERP, there are three different collaboration profiles (developers, functional experts and testers), while in the case of Openbravo POSITION, there are only developers. The members of the user community organise themselves and are distributed into active projects in the community.

**Resources available to the community**

Openbravo offers a range of resources (some in more than one language) for the community and for its partners or general users. These include:

## Recommended website

All of the resources mentioned can be accessed from the company website (http://www.openbravo.com/).

- Corporate website
- Partners area
- wiki project

- Portal for the Openbravo user community
- Employee blogs
- Forge for products (Openbravo ERP and Openbravo POS)
- Bug tracker
- University
- Mailing lists
- Openbravo code repository
- Openbravo news service

The user community can refer generally to a specific guide in the wiki explaining how to collaborate with the project. It also has an exhaustive list of communication channels that it can access. The roadmaps of each product complete the resource guide for the user community.

# Positioning and evolution

The company was founded as Tecnicia in 2001. In 2006, it obtained funding in excess of six million dollars, when it was renamed Openbravo. That same year, it released the source code of the products it develops under free licenses.

In May 2008, the funding round amounted to over twelve million dollars, with investors including Sodena, GIMV, Adara and Amadeus Capital Partners.

Over the years, Openbravo has won several business and free software awards and received grants from the Spanish Ministry of Industry, Tourism and Trade's PROFIT programme to promote technical research.

Both the company and the user community display a positive trend in development, given that the project is currently one of the twenty-five most active on SourceForge with more than one million cumulative downloads in early 2009.

### Recommended website

For the most active projects on SourceForge:

http://sourceforge.net/top/mostactive.php?type=week.

# abstract

In this module, we have described the main features of the creation, management and maintenance of free software development projects, taking into account the participation of the user community.

In a sense, the foundations of free software production do not differ that much from the methodologies of traditional software development. However, the features of open code and the presence of the user community shape its operation, making it unique in many respects.

With regard to the project per se, we must stress the importance of identifying, defining and structuring both the functional aspects of the project (infrastructure, version management and coordination measures) and those concerning free software (credibility, transparency or typologies of participation).

In addition to these aspects, there are factors associated with the free software community, such as the company's strategy for community management (Open Source strategy), the product life cycle and methodology, quality management and the legal aspects of user contributions.

Lastly, we described a case study that is representative of many of the aspects we have seen in the different sections.

# Bibliography

## bibliography

Bain, M. et al.. 2007. *Aspectes legals i d'explotació del programari lliure*. . Universitat Oberta de Catalunya < http://ocw.uoc.edu/informatica-tecnologia-i-multimedia/aspectes-legals-i-dexplotacio/Course_listing > [Consulted in February 2009].

Collins-Sussman, B.; Fitzpatrick B.. 2007. *What's in it for me? How your company can benefit from open sourcing code*. . OSCON: 27 July 2007 < http://www.youtube.com/watch?v=ZtYJoatnHb8 > and slides < http://www.red-bean.com/fitz/presentations/2007-07-27-OSCON-whats-in-it-for-me.pdf > [Consulted in February 2009].

Crowston, K.; Howison, J.. (May 2006).*Assessing the Health of a FLOSS Community*. . IT Systems Perspectives (pp. 113-115). < http://floss.syr.edu/publications/Crowston2006Assessing_the_health_of_open_source_communities.pdf > [Consulted in February 2009].

Fogel, K.. 2005. *Producing Open Source Software: How to Run a Successful Free Software Project*.. < http://producingoss.com [http://producingoss.com/] > [Consulted in February 2009].

Mako, B.. 2001. *Free Software Project Management HOW TO*. . < http://mako.cc/projects/howto > [Consulted in February 2009].

Mohindra, D.. 2008. *Managing Quality in Open Source Software*.. < http://www1.webng.com/dhruv/material/managing_quality_in_oo.pdf > [Consulted in February 2009].

**Openbravo** <http://www.openbravo.com/> [Consulted in February 2009].

Raymond, E.. 1997. *The cathedral and the bazaar*. < http://www.catb.org/~esr/writings/cathedral-bazaar/ > [Consulted in February 2009].

Tawileh, A. et al.. (August 2006).*Managing Quality in the Free and Open Source Software Community* . (pp. 4-6). Proceedings of the Twelfth Americas Conference on Information Systems. Mexico: Acapulco. < http://www.tawileh.net/anas//files/downloads/papers/FOSS-QA.pdf?download > [Consulted in February 2009].

# Chapter 6. Strategies of free software as a business

Amadeu Albós Raya

GNUFDL
2010-02-01

# preface

In the free software business, as with generally any technology-based business, a myriad of factors come into play that can influence the success of the project to varying degrees. Many of these factors, such as the characteristics of the software market, business models or the special features of free software production, are addressed in the other modules.

The series of actions allowing us to establish a business opportunity that is valid and viable in practice must be finely tuned if we are to secure our aims. In other words, it is essential to transfer the features of free software as a business to the real target market in order to implement a specific and appropriate business strategy that can exploit the advantages of free software and control its disadvantages.

This strategy must reflect the reality of the environment and business context, identifying and analysing the points of view of each player on the market, in order to maximise the guarantees of success as much as possible.

In this module, we will describe the main features influencing the strategy of businesses based on free software, characterising the different elements as advantages or disadvantages for the business.

# objectives

After completing this module, students should have achieved the following aims:

1. To understand the importance of strategy in businesses based on free software.
2. To identify and evaluate the advantages of exploiting free software as a business.
3. To identify and evaluate the disadvantages associated with the free software business.
4. To obtain a thorough knowledge of and relate the strategies for free software business models.

# The competitiveness of free software

### Important

Nowadays, free software is a valid and viable alternative to proprietary software. Features such as the modularity of its development and installation, a standard-based operation and the constant evolution of applications form an adequate basis for the competitiveness of free software.

Nonetheless, this competitiveness will not be sufficient for the free software business if these and other features are not properly channelled. In other words, to create a project that will be stable and reliable

over time, we must define a business strategy to unite and coordinate the advantages while managing and controlling the disadvantages.

In this first section, we will look briefly at the main features that make free software a competitive alternative to proprietary software.

## Recommended website

M. Boyer; J. Robert. 2006. *The economics of Free and Open Source Software: Contributions to a Government Policy on Open Source Software*. (Ch. 3, "Advantages and disadvantages of FOSS").

<http://www.cirano.qc.ca/pdf/publication/2006RP-03.pdf>

**Cost**

In general, applications based on free software are freely available at no charge from the Internet. This distribution philosophy is the antipode of the proprietary model, where payment is usually required for limited use of the application in binary format.

Consequently, cost is a significant competitive advantage for the adoption of free software over proprietary alternatives, given that it can substantially reduce the required investment for a technological implementation (whether created from scratch or for a major system overhaul).

The reduction in costs can also be significant in the evolution or specialisation of a particular application because while free software guarantees the possibility of aligning the application with specific interests through free access to the source code, the proprietary equivalent may require a completely new development.

**Development, flexibility and modularity**

While the development of a technological solution based on free software may sometimes differ only slightly from the proprietary equivalent, methodologies based on collaboration and co-evolution between company and user community have the advantage of cooperations of scale.

These features offer a number of possibilities, ranging from the exploitation of economies of scale and the creation of segmented markets to the flexibility and modularity that enhance both the interoperability and integration between applications and their extension and evolution. In short, these features encourage the generation of specific business opportunities.

**Technology risk**

Generally speaking, the risks associated with technology adoption affect free and proprietary software equally, at least from a strictly technological point of view.

As a result, in the case of specific applications or solutions, the risk bears more relation to the specific capabilities and competencies of the latter than to the technology or methodology used for their development.

**Security, reliability and life cycle**

Over time, the evolution of software development methodologies has led to greater and better control of the quality of the software produced, particularly in areas such as adaptation and bug-fixing.

In this case, the opening up of the process of free software development and the collaboration of the user community in the latter affords substantial differentiation from the proprietary model. In other words, it

will be difficult for a company that produces proprietary software to match the human and time resources used in free software projects.

This unique feature of free software adds to the competitiveness and reliability of solutions, both for companies and for their customers.

**Support and documentation**

Occasionally, applications based on free software can lack the packaging we are usually offered by their equivalent proprietary software applications. From a sales point of view, this situation is a source of business opportunities on several levels, with the additional benefits that specialisation and customer proximity can bring.

**Change management**

Free software encourages the restructuring of the values integrated in the traditional market: it provides independence, freedom, lower costs and investment efficiency, many of which have been mitigated in the traditional software business.

### Example 6.1. Restructuring of values

Free software provides independence, freedom, lower costs and efficiency of investments, many of which have been mitigated in the traditional software business.

It also allows companies to adjust the cost structure and establish coopetition strategies with related or complementary providers. This situation is more advantageous, competitive and effective – and less risky – for its participants than their proprietary consortium equivalents.

# The customer perspective

### Important

For customers of products and services based on free software, it is very important to identify the advantages and disadvantages of the free software model in comparison to proprietary formats, especially if the latter takes place in the context of a consolidated traditional market.

From the point of view of software product customers, economic issues may be more relevant to the final implementation of the technology than a technological differentiation in product architecture. These features need to be taken into account in company strategies if we intend to exploit the market successfully.

In the following sections, we will describe in detail the advantages and disadvantages of businesses based on free software from a customer point of view.

# Advantages

The advantages of free software for customers constitute an important part of the company's business opportunity because they affect its market positioning.

**Economic effects**

Free software gives the customer independence from technology providers, alternatives to proprietary products and services (or possibly other free solutions), use of an increasing range of software linked to standards and their subsequent complementarities, and interchangeable software situations (commoditisation).

**Costs**

The increased efficiency and effectiveness in the management of technology costs can be very significant for end customers, whether individuals or companies of any size.

Due to its more efficient and effective management, free software encourages the introduction of changes in cost structure and the technology investments of customers.

### Example 6.2. Changes in costs

We can cut implementation costs by using free software distributed without charge or by reducing the forced upgrading of equipment within very short periods. These savings can then be used to finance services or long-term technology investments (such as lower system maintenance costs).

### Recommended website

*Análisis Financiero del Software Libre* . J. García; A. Romeo; C. Prieto (2003). *Análisis Financiero del Software Libre* (Ch. 7) <http://www.lapastillaroja.net/capitulos_liberados_pdf/la_pastilla_roja_ capitulo_7.pdf [http://www.lapastillaroja.net/capitulos_liberados_pdf/la_pastilla_roja_capitulo_7.pdf]>

In addition, free access to the source code encourages the specialisation and extension of applications based on free software by the customer – or by a specialist company.

**Ethical values**

In some cases, the ethical values associated with the free software movement, such as transparency, independence, equality and cooperation, may be appropriate to the aims and ends of the customer – or to the image it attempts to portray.

# Disadvantages

Despite the obvious benefits of free software for customers, it also has disadvantages that need to be controlled and mitigated by companies seeking to exploit related business opportunities.

**Economic effects**

Customers can be reluctant to embrace free software because of switching costs or compatibility with the solutions that it uses. The evaluation of alternatives can sometimes be biased by the search for short-term results or returns, the technology myths associated with free software or the customer's historical association with the software it uses.

**Risk management**

Any technological implementation in an organisation will have a degree of associated risk (even for private customers), broadly comparable in free and proprietary software. For the customer, the possible nuances between the two solutions may be unsurmountable in certain conditions, such as when the customer has a history of one or more failed migration attempts.

The customer may sometimes be unwilling to take risks with new software that could affect the regular operation of processes, technology and staff, doing away with the need to adapt them to enhance the organisation's efficiency after a relevant technology implementation. This can also be a further source of operational problems if it is not carefully planned.

**Cost management**

Some of the costs of an implementation may be common regardless of whether free or proprietary software is used. Customers sometimes believe that platform changes inevitably involve more costs due to training, support and staff motivation, or due to the loss of company productivity, for example. It can be difficult to counter these arguments, mainly because they are difficult to measure and quantify economically.

# Business strategy

The vision of the customer and, by extension, the target market is essential for defining a sound business strategy for the company. Nonetheless, the company must complete its strategy by taking into account the advantages and disadvantages of the free software model and, more specifically, the particular business model it exploits.

### Important

Companies that commercially exploit free software should be aware of and realistic about the environment in which they operate. All the special features of free software, customers and the business model exploited need to be identified and analysed before it can formalise a realistic and appropriate strategy to secure its aims.

In this section, we will look initially at the advantages and disadvantages of the free software model for business before analysing the strategies associated with business models based on free software.

# The free software model

As is the case with customers, the special features of the free software model influence both the definition of the business and the possibilities of establishing the company on the market and its long-term prospects of business development.

**Advantages**

We will now deal with the main advantages for the provider or company that exploits free software for profit.

- **Positioning and differentiation**

Companies that exploit free software can adopt a good position for positive marketing and market publicity in the sense that the diffusion of free software may promote the aims of consolidation, trust, sustainability and increased popularity of the company.

- **Market**

In the traditional software market, it can be difficult to identify and exploit new business opportunities because of the economic impact of traditional business policies. Therefore, to reiterate what we have explained above, free software encourages the introduction of innovative (disruptive) technologies that allow for a differential bias which can be harnessed for new business opportunities.

Thus, free software favours the penetration of new companies in the traditional market by disrupting the economic effects that immobilise the market players.

- **Development and distribution**

The freedom, ease and low cost of the distribution of free software (usually by free and direct download from the Internet), combined with the cooperation, involvement and motivation of the user community in its development, encourage both the spread and adoption of applications. In other words, both the

development methodology and the special features of the distribution of solutions promote the efficiency and effectiveness of the project.

- **Costs and risks**

The burden and structure of costs and risks of companies based on free software may be more advantageous and competitive than models based on proprietary software because of the distribution and decentralisation of some of its processes among the different players involved.

- **Commoditisation**

The commoditisation of software is advantageous for all players because it reduces the barriers to entry for new software producers and increases the competitiveness of the sector, thereby allowing production of the same goods more efficiently. Besides seeking specialisation and differentiation to exploit business opportunities, it is also possible to do business in a completely commoditised market.

- **Innovation and the creation of value**

## Recommended reading

L. Morgan; P. Finnegan. 2008. *Deciding on open innovation: an exploration of how firms create and capture value with open source software*. (Vol. 287, pp. 229-246). IFIP 2008.

Open and cooperative development and production methodologies result in greater efficiency and effectiveness, both in the creative process of innovation and in the creation and capture of value by the company. In other words, by opening up its production processes, the company ceases to rely solely on internal staff for innovation (which is limited by time and aims) and begins to benefit from the ideas and insights of volunteers, users and customers (whose flexibility and motivation fosters the emergence of interesting innovations).

This closes the feedback loop between the company and customers or users (treated as co-developers), thus reducing project risk and maximising the guarantees of success.

**Disadvantages**

We will now discuss some of the problems that can arise in companies based on free software.

- **Economic effects**

## Example 6.3. Limitations

For example, customer captivity and economy of ideas prevent companies from securing a dominant position on the market, as could occur on certain markets swamped by proprietary solutions.

Some of the economic effects that favour the introduction of a new company on the market could also limit the quality and quantity of its operations.

- **Results**

One consequence of the above is that it can be difficult to make large profits (at least to the degree that proprietary software corporations do today) or profits that can be sustained over a long period of time.

- **Commoditisation**

The commoditisation of software can also have a negative effect on companies based on free software if they fail to adequately identify and plan the differentiation of their products, services and even marketing

policies. In other words, a situation of interchangeable goods can affect the composition and distribution of the market if the products do not provide substantial differentiation over time.

Moreover, doing business on a commoditised market makes it impossible to obtain large profit margins because it is relatively easy for customers to change technology provider. So, a business must be truly better than or at least as good as its competitors in the industry in order to hold on to its position, for example by focusing on response times and ability to adapt.

- **Mythology**

Despite the passing of the years, there may still be some myths about free software on certain markets that complicate its implementation and deployment. The difficulty in debunking these myths will depend on market characteristics such as the degree of implementation of proprietary software or failed attempts at migration to free software.

# Free software production

In general, if the developed application is successful among potential customers, we can obtain advantages in the attraction of improvements and complements, the sympathy of the audience and community, and lower maintenance costs due to the participation of the community.

By contrast, it can be difficult in free software development to recover the initial investment, which can sometimes be quite substantial. While it is a common problem in both free and proprietary software, it is more difficult to sell copies of free software than other models.

**Mixed models**

The duality of mixed models (usually a public and a commercial version) favours the adoption and diffusion of the application but has some drawbacks too, such as the limited involvement of the community in the business aims or the need to maintain an interesting commercial product over time.

This latter aspect may generate other problems if the company's management of the user community is inadequate. For example, the community may develop the proprietary extensions to the commercial version by itself – and publicly.

**Software and services**

For the provision of services associated with a free application, it is possible to develop coopetition strategies to expand the target market, subsequently segmenting through differentiation. If coopetition strategies cannot be established, the model offers few barriers to entry for competitors which, given their access to the source code, can equip themselves with the necessary infrastructure to compete as they would on traditional markets.

Moreover, obtaining a substantial income solely from related services can be difficult in markets with a strong presence of innovators and technology enthusiasts.

# Provision of services related to free software

The provision of services has some advantages over its proprietary software equivalent, such as the absence of substantial licensing costs, product quality and access to the source code. These features allow the efficient and effective provision of services, resulting in significant added value for the customer.

Nonetheless, it can be difficult to hold on to customers in the long run due to the ease of market entry and the difficulty of providers to differentiate their services.

**Small and medium enterprises**

The main business opportunities concern the lack of packaging and the distribution of applications based on free software (such as installation, support, customisation or training), with the exploitation of specific niche markets.

By contrast, custom developments on specific applications may encounter difficulties with integration and compatibility with later versions. The emergence of competitors in the same industry can also be problematic because of the limited scope for coopetition.

**Large companies**

The participation of large companies in projects based on free software can be relatively straightforward due to the existence of a prior infrastructure and organisation. The use of free software also cuts costs and improves brand image in areas such as reliability, strength, confidence, stability and professional support.

Nonetheless, formalising a brand image is not easy in the short term. The dominance of large proprietary software corporations on the market complicates positioning, and the risk associated with big projects is also greater.

# Ancillary markets

In general, the business models associated with ancillary markets can serve to complement main strategies. However, they may be appropriate and viable as a basic strategy in markets with little competition or with differentiation or specialisation requirements.

**Hardware**

The ancillary market of hardware may prove valid for exploiting markets that require a product specialisation, such as integrated services, high performance or lower purchasing costs for customers, i.e. markets in which proprietary systems may have no interest and free software can constitute a significant difference for customers.

The main disadvantages relate to the ability to bear the costs of production and development if the target market is limited or there is strong price competition. The difficulty in recovering the initial investment may sometimes make this market unsuitable for small and medium enterprises.

**Other markets**

Ancillary markets such as the sale of books or merchandising may be competitively equivalent to their proprietary software counterparts given the special features of free software, such as complementarities with the original product or the dissemination of ethical values.

# abstract

The free software model is a valid and viable alternative to proprietary software, formalising competitive features in its implementation with very varied aims, such as cost and flexibility.

These features are advantages and disadvantages for the main players in the software market. Sometimes, aspects can be an advantage for some and a disadvantage for others, which highlights the need to formalise a realistic business strategy that can guarantee aims efficiently and effectively.

To develop this strategy, companies based on free software should consider the implications of the free software model both for the customer and for its own operation:

- Free software allows customers to combat the economic effects of a traditional market and manage the cost of implementation better, at the cost of assuming a degree of risk.

- For the company, it is a business opportunity based on differentiation and the efficient management of costs and risks, at the cost of limiting its market position and the results it could obtain.

Formalisation of its business strategy will allow the company to exploit more and better free software advantages in the context of the company's activity, while also managing and mitigating the disadvantages that may limit its guarantees of success.

# Bibliography

## bibliography

Boyer, M.; Robert, J.. 2006. *The economics of Free and Open Source Software: Contributions to a Government Policy on Open Source Software*. . Centre Interuniversitaire de recherche en analyse des organisations (CIRANO), 2006RP-03 < http://www.cirano.qc.ca/pdf/publication/2006RP-03.pdf > [Consulted in March 2009].

García, J.; Romeo, A.; Prieto, C.. 2003. *Análisis Financiero del Software Libre*. . La Pastilla Roja, Chapter 7.

<http://www.lapastillaroja.net/capitulos_liberados_pdf/la_pastilla_roja_capitulo_7.pdf> [Consulted in March 2009]

Ghosh, R. A. UNU-MERIT, N. L.. 2006. *Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies sector in the EU* .

<http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf> [Consulted in March 2009].

Iansiti, M.; Richards, G. L.. 2006. *The Business of Free Software: Enterprise Incentives, Investment, and Motivation in the Open Source Community*..

<http://www.hbs.edu/research/pdf/07-028.pdf> [Consulted in March 2009].

Morgan, L.; Finnegan, P.. 2008. "Deciding on open innovation: an exploration of how firms create and capture value with open source software". In: G. León; A. Bernardos; J. Casar; K. Kautz; J. de Gross (eds). "International Federation for Information Processing".*Open IT-Based Innovation: Moving Towards Cooperative IT Transfer and Knowledge Diffusion*. (Vol. 287, pp. 229-246). Boston: Springer.

West, J.; Gallagher, S.. 2006. "Patterns of Open Innovation in Open Source Software". In: Henry Chesbrough; Wim Vanhaverbeke; Joel West (eds.).*Open Innovation: Researching a New Paradigm*. (pp. 82-106). Oxford: Oxford University Press.

<http://www.openinnovation.net/Book/NewParadigm/Chapters/index.html> [Consulted in June 2008].

Wheeler, D.. 2007. *Why Open Source Software?*.

<http://www.dwheeler.com/oss_fs_why.html> [Consulted in March 2009]

# Chapter 7. Free software, a new economic model?

Amadeu Albós Raya

GNUFDL
2010-02-01

# preface

This module will examine the paradigm of free software from the point of view of an economic model. In other words, we will study the fit and viability of free software as a model of economic operation that can be sustained in the long term.

In the study of free software as an economic model, we are limited by the relative youth of businesses based on free software. However, considering that the economic rules of the market are generally the same, we will base our study on the differentiation introduced by the free software business with respect to traditional markets. This view will give us a realistic initial approach to the qualities of free software as an economic model.

First of all, we will explore the foundations of the paradigm of free software and, hence, its operation and possibilities. More specifically, we will describe the conceptual features of the underlying operational philosophy of the model, such as its social production.

We will then explore the consequences of the model based on free software from different points of view, taking into account its differences with traditional models of software production and business. The projection of these concepts should give us a better understanding of how the free software model could fit into the market in the near future.

Lastly, we will study how the free software model relates to the validity and viability of companies based on it, explaining the importance of combining strategy with opportunity.

# objectives

After completing this module, students should have achieved the following aims:

1. To be familiar with the economics of the model based on free software.
2. To understand the fundamentals and implications of the free software model vis-à-vis the traditional model.
3. To understand the differentiation introduced by the free software model and evaluate its suitability for the creation of value for the market.
4. To explore the validity and viability of the free software model and exploitable business models.

# Basis of the model

We are familiar with many of the technological features of free software that are similar to those of proprietary software to varying degrees. In other words, the fundamental differences – if any – between free and proprietary software are not based on the internal and external aspects of the product.

Broadly speaking, the technology applied to a product (for example, the design, architecture or specific implementation) does not in itself create a substantial differentiation between free and proprietary models, at least from the strict point of view of the finished product.

### Important

The differences between free software and other paradigms of software production (especially proprietary) mainly concern the specific features of the model of development, the user community and the differentiation of the product's value added.

These differences do not lie in technological aspects unique to the development of the application or software, but on the characteristics and implications underlying their production. In other words, free software sums up a particular orientation to create value in products and services that differ from the traditional point of view.

As explained in previous modules, the business models that exploit these distinctive features in a traditional market have been perfected in recent years. In all events, the chief value lies not in the software itself but in the capital acquired when it is adopted.

This capital formalises the foundations of free software. In other words, free software is based on the social production and network culture that not only allow but also promote its possibilities and effects.

The following sections will succinctly develop these two concepts. We will first of all examine the main features of social production before moving on to characterise network culture and its impact on the economics on which free software is based.

# Social production

Advances in global communications and the democratisation of technology in recent decades could have influenced what we now consider free software in different ways.

That is, the ease of access to information and willingness to cooperate are not unique features of free software; they form a basis for the development of valid and viable alternatives in many fields.

While there are now many initiatives associated to varying degrees with social production, in this model, business organisations discover a way to encourage creation and attract value for their business models.

### Example 7.1. An example of social production

Wikipedia (http://www.wikipedia.org/).

### Recommended reading

L. Morgan; P. Finnegan. 2008. *Deciding on open innovation: an exploration of how firms create and capture value with open source software*. (Vol. 287, pp. 229-246). IFIP.

In *The Wealth of Networks*, Yochai Benkler explores this issue in detail. Below, we will discuss some of the most relevant aspects characterising social production.

### Recommended website

Y. Benkler. 2006. *The Wealth of Networks: How social production transforms markets and freedom.*. ( http://www.benkler.org/Benkler _Wealth_Of_Networks.pdf [http://www.benkler.org/Benkler_Wealth_Of_Networks.pdf] ).

**Economics of information**

Information is a public good with economic implications at different levels as a result of the use of information technologies.

Innovation, as the creation of new information, may be adversely affected by situations with restriction or control, and facilitated by openness and collaboration on the production of information, knowledge and culture.

Hence, production or innovation in peer-to-peer networks or generates a spiral of opportunities characterised by motivation and efficiency with technological support.

### Peer-to-peer networks

In this case, the term refers to the operation of the community, rather than the architectural or technological basis of communication.

**Development and distribution of information**

The development and distribution of information can follow a variety of patterns, depending on how freedom is distributed between producers and consumers. In general, the more freedom given to producers, the less obtained by consumers.

Distribution channels for information influence how the latter is shared. The direction of the transfer and its aims also influence how information is shared.

In all events, licensing and patents can restrict the flow of information, while the quantitative growth of the network need not fragment or restrict it.

**Implications of social production**

Benkler maintains that the way we see the operational structure of the world around us is changing, especially in terms of how we all collaborate and interact with the integration of ideas and knowledge to create new knowledge.

## Networked economy and culture

The implications of social production have become apparent in many fields in recent times, particularly in free software. The interaction of knowledge and the refinement of ideas is now a good way to encourage and further develop a concept.

### Important

This view of production as a collaboration to qualitatively achieve a given aim contrasts with the more traditional view of the market of ideas and knowledge, where the importance lies more with the final adoption of the product than with consensus, fit or quality.

David Bollier's *When Push Comes to Pull: The New Economy and Culture of Networking Technology* explores how the evolution of information technology has allowed a new point of view to emerge that contrasts with the centralisation and hierarchy of the traditional model.

### Recommended website

D. Bollier. 2006. *When Push Comes to Pull: The New Economy and Culture of Networking Technology.* . ( http://www.aspeninstitute.org/atf/cf/%7bDEB6F227-659B-4EC8-8F84-

8DF23CA704F5%7d/2005InfoTechText.pdf [http://www.aspeninstitute.org/atf/cf/%7bDEB6F227-659B-4EC8-8F84-8DF23CA704F5%7d/2005InfoTechText.pdf] ).

The following sections will now briefly examine the main economic and cultural features of networked culture considered by Bollier.

**The push and pull models**

The push model is based on mass production, anticipating consumer demand and dynamically managing time and the location of production resources.

The pull model is based on the openness and flexibility of the production platforms used as resources. This model does not anticipate consumer demand, but rather customises products according to demand using fast and dynamic processes.

**Value creation networks**

In pull models, the sharing of information and best practices substantially improves the corpus of knowledge of all members of the network.

This network promotes and integrates open business models based on the creation of value and product customisation or differentiation.

Hence, pull model platforms formalise, improve and increase the flexibility of innovation and evolution through the community, without incurring the costs of a similar implementation in a push model.

**Target market**

Push models are successful in areas where consumers are not very clear on what they want and prefer to make their selection based on predefined typologies.

By contrast, in pull models, consumers want to form part of the production and selection process, in the sense that they may not know exactly what they want, but they are sure that they want to participate and form part of the process.

**Production**

Push models tend to seek alternative forms of production that may be more economically competitive (for example, lower production costs), while pull models tend mainly to seek the best ways to add value to the production network.

This special orientation of pull models favours the scalability of the production network and the union of the best participants for production specialisation.

**Cooperation**

Pull models favour the creation of relationships based on trust, the sharing of knowledge and cooperation among members of the network, to everybody's benefit.

This ethos is often transformed into a system of collective government for the sustainable and fair management of shared resources.

In this sense, companies based on pull models should provide guarantees for the recognition of network members, since the model is based on trust and the creation of value.

**Education**

With push models, the activity of students is focused on the construction of static knowledge as prior training for a subsequent hierarchical society.

Pull models promote alternative forms of education in that information technologies allow students to enter a dynamic flow of activity with access to many independent resources for creating their own corpus of knowledge (and sharing it).

# Characteristics of the free software model

The foundations of free software formalise a structure in which cooperation and the sharing of knowledge among members allow for the innovation, production and evolution of global knowledge.

The creation of value is undoubtedly an important goal for all members of the community (be they users, developers, etc.) and for the model itself. Hence, the decentralisation, freedom and independence that are the mark of the community offer guarantees for the consolidation and cohesion of production and social capital.

### Important

The free software model is based on differentiation in relation to the values that govern the traditional market, both from the point of view of software development and of appreciation of the value created.

While it is true that, from a traditional point of view, some of the features of the free software model are also applicable to other paradigms of development and value creation, the free software model introduces new features to the perception and appreciation of the values associated with the traditional market.

In this section of the module, we will determine the features of the free software model by comparing them with those of a traditional model, with the aim of assessing the real differentiation introduced by the model in daily practice.

First of all, we will discuss the model from the point of view of software development, before moving on to analyse the implications of differentiation as a paradigm based on social production.

# Software development

The methodology of free software development is possibly one of the factors popularly considered as a differentiation compared to other software development paradigms, such as the proprietary model. But is this really the case?

### Important

From the point of view of software production, there are points in free software development that clearly overlap with other development models, such as proprietary, since the production methodologies have a certain independence from specific implementations.

However, the fact that software production may be more or less consistent with other models or that some of the requirements for code freedom are more or less necessary in practice, this does not mean that there cannot be significant differences in other aspects leading us to evaluate the whole as innovative.

Fuggetta's article *Software libre y de código abierto: ¿un nuevo modelo para el desarrollo de software?* explores these and other aspects of the differences between the development model of free software and the development model of proprietary software. The following sections will briefly outline some of its findings.

## Recommended website

A. Fuggetta. 2004. *Software libre y de código abierto: ¿un nuevo modelo para el desarrollo de software?*. ( http://alarcos.inf-cr.uclm.es/doc/ig1/doc/temas/4/IG1-t4slibreabierto.pdf )

**Context**

The success of free software can be attributed to a range of technological and economic aspects affecting its innovation and production.

Its decentralisation, cooperation and freedom of use and exploitation have made free software the standard-bearer of a new philosophy for addressing and solving a variety of problems.

According to Fuggetta, many beliefs on free software can also be applied to proprietary software, so it is a good idea to explore the topic thoroughly.

**The development process**

From a technological standpoint, the development of free software is not a new paradigm, since most projects have a limited number of collaborators. Moreover, incremental and evolutionary development methodologies are not unique to free software.

Nonetheless, free software has managed to motivate both developers and users to get involved in the project, sharing and associating the development and evolution of the software with the needs of the community.

**Defence of customer rights**

Problems related to customer protection arise mainly in reference to software packages, since the customer already owns the code in custom developments.

For software packages, it may be enough to be able to access the source code without subsequently modifying or redistributing it. The company's user support should also abide by rules that facilitate the handing over of the code in the event that the company cannot maintain it.

**Dissemination of knowledge**

The spread of knowledge through access to the source code is insufficient, since the subjects on software engineering reveal that documents describing the software architecture are also needed.

Moreover, in the event that this knowledge could be disseminated, it would only be necessary to publish its source code (without the right to copy and redistribute the software).

**Cost**

The fact that the software is released under a free license does not mean it cannot be commercialised or that its development does not have an associated cost (although we do not know the extent of this).

In addition, just because we cannot quantify or centralise its cost, this does not mean that it is not distributed among the collaborators, even indirectly by companies with little or nothing in common with the world of software.

**Effectiveness of the business model**

The main business models that actually exploit free software engage in the development and distribution of pure open source packages or free and proprietary software platforms. Other forms of business can be set up to a greater or lesser extent with both free and proprietary software.

Moreover, there is currently no evidence to suggest that a company based solely on services will be profitable over time.

### On profitability

In *The Business of Software*, Michael Cusumano argues that software companies will increasingly depend on the combination of revenue from licences and services.

**The software industry**

Europe does not have an industrial strategy to unify the actions of the various companies involved. Hence, offering support to free software is not a strategy in comparison to the creation of innovative products.

# The cooperative paradigm

While some of the features of the free software model are not innovative from a classical perspective, as we saw above, those that motivate a change in market perspective are.

### Important

To analyse in detail the differentiation introduced by the free software model compared to other traditional models, we will need to assess the aspects of production and the creation of value and knowledge underpinning the model.

In his article *Open Source Paradigm Shift*, Tim O'Reilly identifies these and other features of free software that are differentiating and which create a competitive advantage that can be exploited for profit. The following paragraphs will briefly outline some of his findings.

### Recommended website

T. O'Reilly. 2004. *Open Source Paradigm Shift.*.

(http://www.oreillynet.com/pub/a/oreilly/tim/articles/paradigmshift_0504.html).

**Change**

Free software has deeply transformed the structure of the benchmark market, often with implications extending beyond those imagined by its creators.

These changes are based on product quality, lower production costs and the use of standards, in addition to differentiation in marketing, distribution and logistics.

**Software as a commodity**

In a context of permanent standardised communications such as the current one, all communication applications are interchangeable (a web browser, for example). In other words, the use of standards means that software can be considered a commodity.

Hence, when the revenue-generating potential of an application is diminished because of the commoditisation process, a new market will emerge for proprietary products, especially if they exploit the global communications network.

Moreover, free software remains a viable model for companies providing services, although we cannot expect similar profit margins to those of the modern software giants.

**Network collaboration**

The culture of software sharing has grown since its origins at the same pace as the Internet, whose participatory architecture is present in virtually all of its functionalities.

Free software is the natural language of the networked community, resulting in a style of collaboration and participation unique to its members. This collaboration is critical to the success and differentiation of leading Internet applications, since it has highlighted the importance of treating users as co-developers of the software.

**Customisation and software as a service**

Nowadays, we are used to considering applications as artefacts rather than static processes. Programs require engineering for their creation but the dynamic languages that allow for the cohesion of components (such as data management scripts) offer the perspective of a dynamic and evolving process of the application.

Many of the services offered on-line (such as search engines) require constant revisions and updates in order to perform their functions properly. This generates a new business paradigm for computers and information technology in general, and for the exploitation of software as a service in particular.

**The Internet operating system**

We can consider the Internet as a single virtual computer that builds an operating system from the connection of several small pieces and allows anybody to participate in the creation of value.

The values of the free software user community are important to the paradigm as they promote the spirit of seeking out and sharing knowledge.

The commoditising of technology is part of the process that allows the industry to move forward to create more value for everybody. For industry, it is essential to strike a balance that will create more value than that obtained with individual participants.

# Validity and feasibility of the free software model

In previous sections, we looked at the foundations of the free software model and the features that distinguish it from the more traditional models.

To evaluate the long-term sustainability of the free software model, we need much more data than we currently possess, i.e. a much wider time slot for a more precise comparison with traditional models.

## Important

Time will tell whether free software is a new economic model and what features and conditions will allow it to be so.

We will now offer some conclusions. Although, at the time of writing, business based on free software is still relatively new, we have highlighted the differences allowing the adoption of a new business perspective based primarily on promoting the cooperative production of knowledge.

**Applications based on free software**

The social production of a specific application or solution encourages the creation of value above and beyond its cost of production, affording it a competitive advantage over other market alternatives.

Free software-based applications, together with open standards, can offset some of the economic effects that strengthen products based on the traditional model. Thus, besides inducing a substantial differentiation with traditional applications, they allow for strategies and policies of coopetition between companies in a win-win paradigm.

**The market**

Social production has plagued the Internet with alternative initiatives to traditional models. Over time, social capital has become a significant value for innovation and development in open environments. We now have profitable business models that pay for the production of knowledge.

## Example 7.2. The business of knowledge

Innocentive (http://www.innocentive.com/) is just one website that rewards ideas that solve specific problems. On it, there are users who pose questions (seekers) and others that solve them (solvers) in exchange for a financial reward.

This and other examples have led to the creation of a new market logic, referred to in some contexts as Wikinomics and crowdsourcing. This logic is based on the pull model we saw above, i.e. the attraction of ideas and effort in contrast to the traditional push model.

In time, we will discover whether this market perspective allows the patterns of technology adoption typical of the traditional market to evolve towards a new situation.

**The business**

The new market perspective can offer new business opportunities associated with the exploitation of ideas, concepts and knowledge for profit without owning the latter. In other words, the value of an application based on free software does not lie in the solution itself but in the capital acquired and generated with it.

Nonetheless, the validity and viability of free software as a model also depends on the particular design features of the company that exploits it. That is, it is essential to design the company around a solid and lasting business opportunity.

**Risks**

Undoubtedly, the main risks for the model based on free software are obtaining a critical mass of users to ensure the project's viability and laying the foundations for a business model that will prove stable over time. We must also take into account the relationship between the initial investment and the expected benefits.

**Business viability study**

Comprehensively analysing, designing and formalising the company will increase the guarantees of success of our free software-based business. To maximise these guarantees, the company's viability must be studied prior to its launch and formalised in a business plan.

### Tip

In the third module of this subject, we took an initial approach to the main features affecting the business viability of the traditional software business, namely aspects of sales and marketing, along with the products and services covered by the business.

Companies based on free software must complement the above aspects with the features of business models based on free software seen in the fourth module, creating a combination to formalise a sound basis on which to set up a sustainable business .

**The free software company**

As with any business model, a company based on free software will also require detailed planning and design prior to start-up. In the previous sections, we emphasised the importance of carefully analysing the business fundamentals of the company as a condition for evaluating its validity and feasibility.

Both the basic features of free software and the implications that we have described throughout this module can exert different influences depending on the typology of the business opportunity and the context we seek to exploit.

Thus, the strategy of a company based on free software can and should characterise its actions in the differentiation of its business and the economic effects of its environment, as well as in social capital and production, in addition to coopetition.

# abstract

Throughout this module, we have explored the features of free software as an economic model, even considering the constraints of data limitations given the fact that business models based on free software are relatively new.

Firstly, the basics of social capital and the collective production of knowledge and ideas are not unique to free software. There are currently several initiatives demonstrating how cooperation and collaboration can be feasible in the innovation and production of knowledge.

These basics reveal the importance of the network of collaborators and their involvement and motivation in the global and individual progress of the members of the community. They also offer a viable alternative to traditional production models.

The implications of the philosophy of social production can also be explored from different points of view. While certain features of free software do not reveal major differences with other models, there are some features that can lead to important distinctions.

In the free software business, it is essential to strengthen and exploit the distinguishing fundamentals of free software in order to provide valid and viable alternatives to traditional models. These actions must inevitably be complemented by the detailed study and planning of the business opportunity to ensure the viability and future of the free software company.

# annex

**GNU GENERAL PUBLIC LICENSE**

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

**Preamble**

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom

to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

**TERMS AND CONDITIONS**

**0. Definitions.**

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

**1. Source Code.**

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

**2. Basic Permissions.**

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose

of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

**3. Protecting Users' Legal Rights From Anti-Circumvention Law.**

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

**4. Conveying Verbatim Copies.**

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

**5. Conveying Modified Source Versions.**

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what

the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

**6. Conveying Non-Source Forms.**

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

**7. Additional Terms.**

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by

this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

**8. Termination.**

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

**9. Acceptance Not Required for Having Copies.**

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

**10. Automatic Licensing of Downstream Recipients.**

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights

granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

**11. Patents.**

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

**12. No Surrender of Others' Freedom.**

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

**13. Use with the GNU Affero General Public License.**

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

**14. Revised Versions of this License.**

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

**15. Disclaimer of Warranty.**

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

**16. Limitation of Liability.**

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER

PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**17. Interpretation of Sections 15 and 16.**

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

**How to Apply These Terms to Your New Programs**

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary

applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

# Bibliography

## bibliography

Benkler, Y.. 2006. *The wealth of networks: How social production reforms markets and freedom*. . New Haven: Yale University Press. < http://www.benkler.org/Benkler_Wealth_Of_Networks.pdf ; WIKI: http://cyber.law.harvard.edu/wealth_of_networks/Main_Page > [Consulted in March 2009]

Bollier, D.. 2006. *When Push Comes to Pull: The New Economy and Culture of Networking Technology*.. < http://www.aspeninstitute.org/atf/cf/%7bDEB6F227-659B-4EC8-8F84-8DF23CA704F5%7d/2005InfoTechText.pdf >

Fogel, K.. 2004. *The Promise of the Post-Copyright World*. < http://www.questioncopyright.org/promise > [Consulted in March 2009]

Fuggetta, A.. (Sept – Oct 2004):*Open Source and Free Software: A New Model for the Software Development Process?*. (No. 171). Novática – Upgrade: Monografía del proceso de software, English < http://www.upgrade-cepis.org/issues/2004/5/up5-5Fuggetta.pdf ) | Spanish: ( http://alarcos.inf-cr.uclm.es/doc/ig1/doc/temas/4/IG1-t4slibreabierto.pdf > [Consulted in February 2009]

Goldhaber, M.. (June 2006).*The Value of Openness in an Attention Economy*. (Vol. 11, no. 6). < http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1334/1254 > [Consulted in March 2009]

Moglen, E.. 1999. *Anarchism Triumphant and the Death of Copyright*. . < http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/684/594 > [Consulted in March 2009]

Morgan, L.; Finnegan, P.. 2008. *Deciding on open innovation: an exploration of how firms create and capture value with open source software*. . In: G. León; A. Bernardos; J. Casar; K. Kautz; J. DeGross (ed.). International Federation for Information Processing. Open IT-Based Innovation: Moving Towards Cooperative IT Transfer and Knowledge Diffusion (Vol. 287, pp. 229-246). Boston: Springer.

O'Reilly, T.. 2004. *Open Source Paradigm Shift*.. < http://tim.oreilly.com/articles/paradigmshift_0504.html > [Consulted in February 2009]

# Bibliography

## bibliography

Sieber, S.. 2006. *Factores que influyen en la adopción del código abierto.*. IESE < www.iese.edu/es/files/Art_Computing_Sieber_Factorescódigoabierto_May06_tcm5-5510.pdf [http://www.iese.edu/es/files/Art_Computing_Sieber_Factorescódigoabierto_May06_tcm5-5510.pdf] > [Consulted in March 2009]

Sieber, S.; Valor, J.. 2005. *Criterios de adopción de las tecnologías de información y comunicación.*. IESE < www.iese.edu/en/files/6_15211.pdf [http://www.iese.edu/en/files/6_15211.pdf] > [Consulted in March 2009]

Dixon, J.. 2007. "The Beekeeper: Crossing the Chasm Between the Cathedral and the Bazaar".*A Description of Professional Open Source Business Models.*. < http://wiki.pentaho.com/pages/viewpageattachments.action?pageId=8346 > [Consulted in March 2009]

Woods, D.; Guliani, G.. 2005. *Open Source for the Enterprise: Managing Risks, Reaping Rewards O'Reilly Media.*. < http://books.google.com/books?id=f4qJiK-ytvgC&dq=open+source+for+the+enterprise >