FINAL REPORT.

RNTL project. “New Economic Models, New Software Industry Economy”.

Managed by Nicolas JULLIEN, Mélanie CLÉMENT-FONTAINE, Jean-Michel DALLE.

Realized thanks to the financial support from *the French National Network for Software Technologies.*

RNTL

Contents

List of Figures

List of Tables

The economic and legal context of a new software economy.

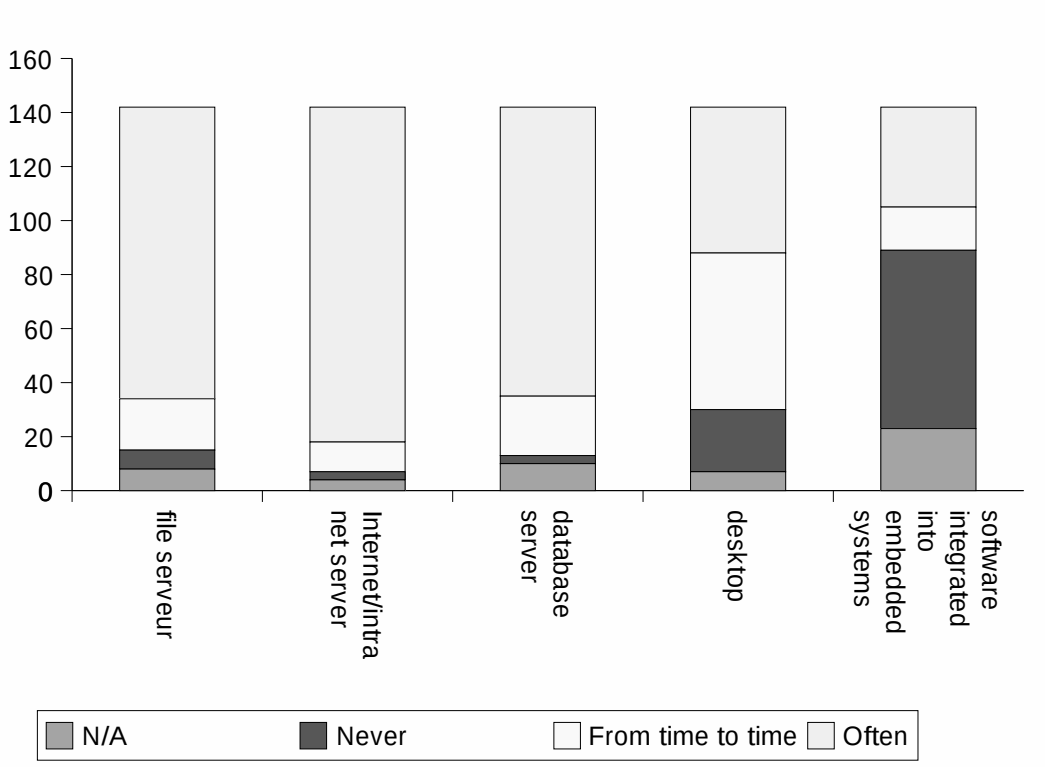
NICOLAS JULLIEN, ENST Bretagne, Nicolas.Jullien@enst-bretagne.fr,
MÉLANIE CLÉMENT-FONTAINE, Alcôve, Université de Montpellier I/ERCIM, melanie@amberlab.net.

This introductory chapter has two objectives: to put the free software production system back in the context of the evolution of information technology in order to explain its emergence and its success today and, secondly, to propose a synthetic analysis of its possible evolution. This analysis will subsequently be developed more thoroughly in the various contributions of this report.
We will devote the first part to the conditions under which free software have emerged. This in turn will lead us to what is at the heart of this report: the study of the strategies pursued by both the users and free software producers. Lastly, we will envisage the legal context in which free software licenses have emerged.

1.1 Free software: a driving force.

First and foremost, the present success of free software seems to have come at the right time as it coincides with the diffusion of internet[1]. Today, commercial offers that are based on free software still involve intranet/internet sites rather than file servers, data bases or client workstations. However, the graph below[2] shows that the latter type of diffusion is now under way.

Figure 1.1 — Solutions built using Free Sofware.



Even if free software programs have been diffused, it would be risky to conclude that the co-operative

[1] The http://www.netaction.org/articles/freesoft.html website reminds us of the importance free software programs have in the way internet functions nowadays. See also Jullien [1999], 2001 on the link between the diffusion of internet and free software programs.

[2] This graph, as well as the following ones, is taken from a survey that Nicolas Jullien made in the French firms which propose products (goods or services) based on free software programs (it was carried out between May and the beginning of July 2002, 141 answers). The data is being analyzed right now and the results is published on the http://www-eco.enst-bretagne.fr/Etudes_projets/RNTL/ site.

organization of free production has been adopted: the precedent of Unix operating system, which was initially developed in an open, co-operative way before being privatized and made available in various "proprietary" versions, is a reminder of this fact. That is why we believe that a more detailed analysis can prove quite useful in order to explain why the diffusion of free software programs can be considered as the new basis for the industrial production of software programs. This analysis will deal with all the characteristics of the previous changes that occurred within the industrial structure in the IT field.

1.1.1 The evolution of the IT production system.

In the field of information technology[3], the industrial structure has undergone major changes. This has happened in two phases: initially and following the arrival of the 360 series, independent hardware and software suppliers emerged. Then, with the arrival of the PC, the operating systems and machines were produced separately. The major conceptual progress in computer design does not explain this evolution: in fact, it remains founded on the principles posed by Von Neumann for the construction of the first computer. This evolution is essentially due to the progress that has been made regarding the basic components, primarily their miniaturization[4].

This progress had two consequences which, when combined, explains the evolution of the software industry. First of all, the price of the machines went down and, as a consequence, they became more affordable to all. Secondly, because machines became more powerful, because software technologies became more and more sophisticated, it was then possible to link, in a more complex manner, the software program (allowing it to fulfill a demand) and the machine on which it was carried out.

Vertical disintegrations and the evolution of the "software" technologies.

Starting in the 1950s, compilers made it possible to separate the way instructions were coded for the machine from the way they were expressed by the computer programmer. Henceforth, a program could be carried out for one machine and re-used on another one, as long as a compiler was adapted to this new machine. The creation of standard programming languages (LISP, FORTRAN) initiated the idea that software programs could be built independently of the machine. These programs were then regarded as products, which was not the case in the beginning. But it was with the invention of the concept of the operating system that one could definitely separate the production of the machine from that of a utilization software program itself. Indeed, one could now make his/her software program evolve, increase its functions, while being certain to find a more powerful machine that could make it work. The operating system had become a standard shared by a whole range of computers. This was the first possibility of vertical disintegration of the computer industry, between the machine on the one hand and the utilization software program on the other.

In the early 1980s, the second possibility was opened thanks to the micro-computer. It grouped together all the elements which made the calculations in a computer. It thus made it possible to separate the production of this component and that of the operating system since the instructions the operating system could send to this computation center were, in turn, standardized. Once again, it became possible to separate the production of the hardware part of the machine from that of the software program (the operating system) which made

[3] Our understanding of the history of the information technology owes much to the work of Breton [1990], Genthon [1995] and Dréan [1996]. The analysis of the organization of the IT industry is also based on the work of Gérard-Varet and Zimmermann [1985], introduced in Delapierre et al. [1980]. Their theoretical point of view was specified by Zimmermann [1989], 1995a and the industrial analysis was presented by Delapierre and Zimmermann [1994]. Lastly, our analysis of the economy and industry of software programs owes much to Horn [2000b], whose work, it seems to us, is a reference in this field. We encourage all those who are eager to know more about these subjects to read these studies.

[4] From vacuum lamps, to transistors, to technologies which are specific to the IT industry, printed circuits and then, thanks to miniaturization, integrated circuits.

it possible to run the whole structure. It became possible to have several hardware producers for the same operating software program.

The separation of industrial activities did follow this technological "disintegration" because some producers (IBM in both cases) brought it on by proposing some new production models along with their new products (the 360 series and the PC)[5].

But this strategic positioning of the producers does not explain, on its own, the evolution one can note in the way software programs are now perceived. They used to be considered a component of the machine that people would readily trade. They have now turned out to be an industrial product protected by very strict utilization licenses. After all, it is those people who produced software programs in a co-operative way, who also invented the operating system or the micro processor. In addition, separating the production of software programs from that of the machines made it easier to re-use them. That should have favored exchanges and co-operation, as it used to be at the earliest stages of information technology.

As far as we are concerned, it is the way users have evolved, and consequently their goals, which explains the success of these new industrial offers.

Industrial organization models.

The fact that the information technology tool is a complex assembling process causes a certain degree of inequality between the users: some are able to understand how it is structured while others only know how to use it.

However, because of standardization, the choice of the users has an influence, both on the type of offer that is available and on the demand of the other users. For example, the greater the number of people who choose an operating system, the wider the range of software programs for this very system. Consequently, it clearly becomes more interesting to choose it. The order in which the adopters choose these technologies is significant. It has an influence on the offer that is finally available and, therefore, on the industrial models which support them. Of course, and as we said earlier in the introduction, this also has a strong impact on the producers' strategies since they seek to impose their offer as *the* standard. Moreover, we did stress that the evolution of industrial organizations was due, first and foremost, to the fact that the latter are the ones who come up with and build new strategies and production systems. But, it clearly is because users have evolved, thus creating needs, that these new organizations impose themselves.

Thus, in the early days of information technology, clients were often military people and/or researchers, who signed contracts similar to research contracts. All of these users had a very high level of technical expertise. Offers were evaluated according to the level of equipment and technological components. The relationship between the client and the provider was one of service. They would co-define their needs and co-construct the answer, in a prototype culture.

But the diffusion of these machines in companies modified constraints. Criteria such as how to recoup the price of these software programs over a longer period of time and, as a consequence, how to re-use them and make them evolve, were considered. The possibility of gaining more knowledge and of constantly developing new functionalities made these software programs increasingly significant for the firms, but also increasingly complex. They required even greater expertise from the developers while, at the same time, users had less and less technical knowledge about the components used in the machines. At the same time, public expenditure decreased and, along with it, the power users and researchers had in this field, since they were the key players. One thus better understands the significance of an industrial model based on a range of machines that would use the same operating system and ignore the subjacent physical architecture. This model would also be based

[5] To realize the importance of these industrial initiatives, all that needs to be done is to remember (cf. Genthon [1995], that the separation between personal computer producers and operating system producers is not a universal standard: it does not exist in Japan and APPLE or SUN continue to design their micro processors and produce their own operating systems.

on the creation of firms which would specialize in the development of application software programs which were specific to firms. And this is, indeed, the model that came to dominate during the 1970s. We went from **a software program which is the component of the machine to a software program which is a customized product for the firm.**

Lastly, the third period to be discussed here is that of the micro information technology, which began in the middle of the 1970s. The idea was to build individual machines which would allow users to free themselves from the often mighty power of "information technology management". The success of IBM PCs is similar to that of its 360 series. Each of the two key elements of the machine, the microprocessor and, more particularly, the operating system, were quickly produced by one firm only which had a quasi-monopoly over the industry. This was the condition required to guarantee the long-term survival of investments and to easily compare the offers of different computers. The specificity of this diffusion lay in the fact that several firms could produce the same machines. This, in turn, created strong competition with regard to equipment and allowed a faster evolution of the quality-price ratio than in the old system. Furthermore, these machines were diffused to meet new uses of personal data processing (spreadsheets and word processing). Lastly, firms were the first to buy these machines, before they became used in offices and, once their price had gone down, by private individuals. The fact that these machines can be used in a non-specific manner and the fact that competition is primarily based on prices, account for the development of software packages (Mowery 1996). This, and its corollary, the tendency for monopolies to arise, intensified in the course of time, as beginner users[6] became prescribers themselves. Indeed, their knowledge, in addition to the brand they had chosen, was limited to the functionalities and ergonomics of the product. It became quite expensive to replace this. From being **specific-products**, software programs became **standard-products**. As a consequence, the offer was quickly standardized around some products according to their function because of the particular characteristics of the software programs. It does not cost anything to reproduce them. Therefore, the more they are used by a large number of people, the less users have to pay development costs.

The mechanisms of evolution.

As a partial conclusion, one can thus say that, thanks to technological progress, more and more users have had access to computers. These users, being less and less qualified in information technology, have needed third parties to develop the software programs which meet their requirements in terms of utilization technologies. The arrival of new users has led to the selection of the models. Industrial positioning has made buying a computer easier than ever before. Simplicity became synonymous with development and sale of standard software programs by producers mainly because these users did not have the competence to develop and evaluate them on their own. One consequence of this progressive dependence on commercialization has been that co-operative production has become more and more marginal.

After this analysis, the co-operative organization of free software program production which primarily relies on technical bases, seems all the more improbable and even anachronistic. Yet, it has shown that, with the increase in demand, new production organizations can emerge. In addition, co-operative production still exists and brings new technologies and new uses to both producers and computer users (Unix, internet and its services, to mention only a few). Once again, it is because technologies and demands have evolved that this organization now proposes a coherent and interesting solution for an increasingly large number of people.

[6] "Naïve" users, to use Delapierre and Zimmermann [1994]'s terminology.

1.1.2 Free software: a new advance in information technology?

Technological progress and customized evolution.

During the 1990s, with the arrival of Internet, the principal technical evolution in information technology was, of course, the generalization of the networking of computers, both inside and outside organizations. Once again, miniaturization allowed the appearance of a new range of products, "nomads" ("organizers" (Psion and Palm), mobile phones). This falls in the constant evolution of information technology products. One has gone from a single machine, dedicated to one task known in advance and reserved for the entire organization, to multiple, linked machines which are used to carry out different tasks, varying in time, and which are integrated in an organization. Networking, exchanging between heterogeneous systems, communication between these machines has become crucial, more especially since they have become more and more numerous. In order to do so, it is necessary to develop standards at the exchange network level. This also needs to be done at the data exchange level, so that they can be transmitted and decipherable by all these systems. Thus, what internet did was not to offer a "protocol" in order to allow the simple transmission of data since this already existed, but to offer a sufficiently simple and flexible one that allowed it to impose itself as a standard for exchange. Software program technologies have evolved too (Horn [2000b]: 126-128): the arrival of object programming languages (Smaltalk, C++, Java and perhaps soon C#) has allowed already developed software components to be re-used. This has led to the concept of "modular software programs": the idea is to develop an ensemble of small software programs (modules or software components), which would each have a specific function. They could be associated with and usable on any machine since their communication interfaces would be standard. Re-using software programs is made easier today by the most recent progress in the field of software engineering (Beugnard [2001]), which makes it possible to better specify the functioning and exchange protocol between components.

What characterizes the technological evolution of software is thus the increasing interdependence between software programs, while the software components that are re-used are becoming increasingly refined and specialized (Zimmermann [1998]). This system can function only if components are indeed re-usable, that is to say, if producers agree on a mechanism which makes it possible to standardize interfaces and guarantee, in time, the stability of these standards. This evolution definitely has consequences on the characteristics of the demand.

User networking has highlighted a greater need for software programs that can exchange data. Indeed, these programs are often produced by different firms, for different users. It is also necessary for these firms to guarantee its availability in terms of time, in spite of changing versions. It thus becomes necessary to standardize the exchange format of the files. It then becomes logical that firms should seek more open solutions which would guarantee them greater control.

The next step is to have adaptable software programs. Indeed, within firms, the demand has become more and more heterogeneous with the networking of various systems and the need for users working in the firm to share the same tools. Software programs (and more particularly, software packages) have to be adapted to the needs and knowledge of every individual.

Final point: the setting-up of the groups of users has also evolved, primarily for reasons of cost. Users who have strong technical skills are using software packages more and more and thus raise the average level of requirement with respect to these products. With use, beginners also realized the inadequacy that existed between some of these software programs and their personal needs. For example, one consequence of the significant number of useless functionalities is that software programs are very "greedy" as far as calculations are concerned. They do not work or, at least, not well enough, on older computers and become quite complex to use.

Once again, the outcome is that commercial relationships are evolving. They have become a **service type**

of relationship, based on the adaptation of standard-component software programs[7]. This is so much the case that Horn [2000b] defends the idea that we would have entered a new phase in the production: "mass custom-made production". However, these service relationships have not proved efficient enough. When one looks into the satisfaction surveys that have been done with regard to information technology products[8], one notes that people are satisfied with respect to the computer itself but not with the after-sales service, especially with software programs. The basic tendency shows that the client "seeks a better before and after-sales support. He/she also wants to be helped to solve his/her difficulties and wants his/her needs to be satisfied".

Free software programs seem to be an answer to the problems met by the software packages in term of quality and interoperability and is developed by some users. Free software also seems to improve service relationships.

Free software programs: improving the quality and respecting standards.

More than mere public research products, free software programs were, first and foremost, tools developed by user-experts, to meet their own needs. The low quality of closed software packages and, especially, the difficulty of making them evolve was one of the fundamental reasons for Richard Stallman's initiative[9]. They are behind many software programs (among which are Linux, Apache or Samba) and have improved them. One must also note that, concerning these flagship software programs, this organization[10] has obtained remarkable results in term of quality and quick improvements[11].

It is undoubtedly due to the free availability of the sources. Indeed, this allowed users to test the software programs, to study their code and correct it if they found errors. The higher the number of contributors, the greater the chance that one of these contributors will find an error, and will know how to correct it. But free software programs are also tools (languages) and programming rules that make this reading possible[12]. All this contributes to guarantee minimum thresholds of robustness for the software.

Co-operative work, the fact that the software programs are often an collection of simultaneously evolving small-scale projects also requires that communication interface should be made public and standardized. Open codes do facilitate the checking of this compatibility and, if need be, the modification of the software programs. By the way, it is also remarkable to note that, in order to avoid the reproduction of diverging versions of Unix, computer firms have set up organizations which must guarantee the compatibility of the various versions and distribution of Linux. They must also publish some technical recommendations on how to program the

[7] The share for IBM service turnover jumped from 32 % to nearly 41 % in four years (1997—2001, see http://www.ibm.com/annualreport/2001/home/index.html). This is the only area that has progressed in such a significant way. Not long ago, HP tried to repurchase the consulting part of PriceWaterhouseCoopers, which IBM finally acquired. Compaq, since repurchased by HP, had announced in June 2001 that it was giving itself 18 months to become a "service company", without mentioning Microsoft, which, with its new license offer and its .Net, continues in its attempts to enter this market.

[8] We are referring here to the satisfaction survey which is carried out each year by 01 IT weekly magazine in French large organizations (numb. 1521 in 1998, 1566 in 1999 and 1612 in 2000). Other inquiries exist which, sometimes, are even harsher, like those of De Bandt [1995], or Dréan [1996] (p. 276 and following).

[9] Who "invented" the concept of free software program, with the creation of the GNU/GPL license and who created the Free Software Foundation, the organization which produces them; see http://www.fsf.org/gnu/thegnuproject.html.

[10] About the way the FSF functions, besides Raymond [1998a], 1998b, 1999, one can also refer to Lakhani and von Hippel [2000] and Jullien [2001].

[11] The Linux users "community" received, 2 years in a row, the best technical assistance award (see http://www.infoworld.com/cgi-bin/displayArchive.pl?/98/05/poy6a.dat.htm). As for performance tests, one can refer to: http://www.syscontrol.ch/e/news/Serversoftware.html for Web servers and http://gnet.dhs.org/stories/bloor.php3 for operating systems. The results for numerous comparative evaluations are available on the following sites http://www.spec.org and http://www.kegel.com/nt-linux-benchmarks.html (the latter mainly deals with NT/Linux). Lastly, Wheeler [2001] has provided a long list of data (evaluation, market shares, etc.) on free software programs.

[12] Let us remember that the first free software programs that were created and adopted were development tools (in particular, compilers, interpreters and programming assistance tools). In fact, software programs allowed the normalization of computer languages, which are the minimal common base necessary to communicate.

See http://www.gnu.org/prep/standards.html for technical recommendations on how to program GNU software.

applications so that they can work with this system in the same spirit as the POSIX standard[13].

In connection with the way one can ensure the interoperability of software programs, this led Zimmermann [1999] to say that, "if the solution of the open systems has suffered because of a recent return to the more isolated proprietary systems, the solution offered by free software programs could contribute to go beyond the limits and contradictions of the systems of intellectual property. It might even lead to a thorough reconfiguration of the software industry".

This is a powerful, high-performance system with regard to the production of software programs. Is this true with regard to the adequacy of the software programs that are produced and the needs of the users? In order for this system to be a real production organization, it must also be adapted to the needs and requirements of the users (and, in particular, to those of businesses, since, historically, they have been the key link in the evolution of the productive model). Users are not always competent enough to evaluate, install or follow the evolution of these software programs. They do not always know how to adapt them to their own needs.

Free software programs: an industrial organization for service production.

It is not because a software program is freely available that it is accessible to everybody. It is necessary to define one's needs, to find a software program that answers them, to install it and, sometimes, to adapt it by developing complementary modules. Once installed, it is necessary to follow its evolution. All this requires the presence of specialists in these software programs which is not always easy. In addition, users, and most particularly businesses, do not need them on a full-time basis. That is why companies should be created and that this activity should be profitable. First of all, however, it is necessary that these offers be more interesting than the already existing solutions which are generally based on closed software programs. Lastly, so as not to break the dynamics of innovation which exist in the software industry, it is necessary to find incentives so that commercial producers can contribute to existing projects and initiate new ones.

Of course, the absence of license fees definitely bestows a competitive advantage on the free solution. But this alone does not justify its adoption: over the long term, this solution must prove to be less expensive and yet offer the same quality standards. Proprietary solution manufacturers use this indicator to defend their offers[14].

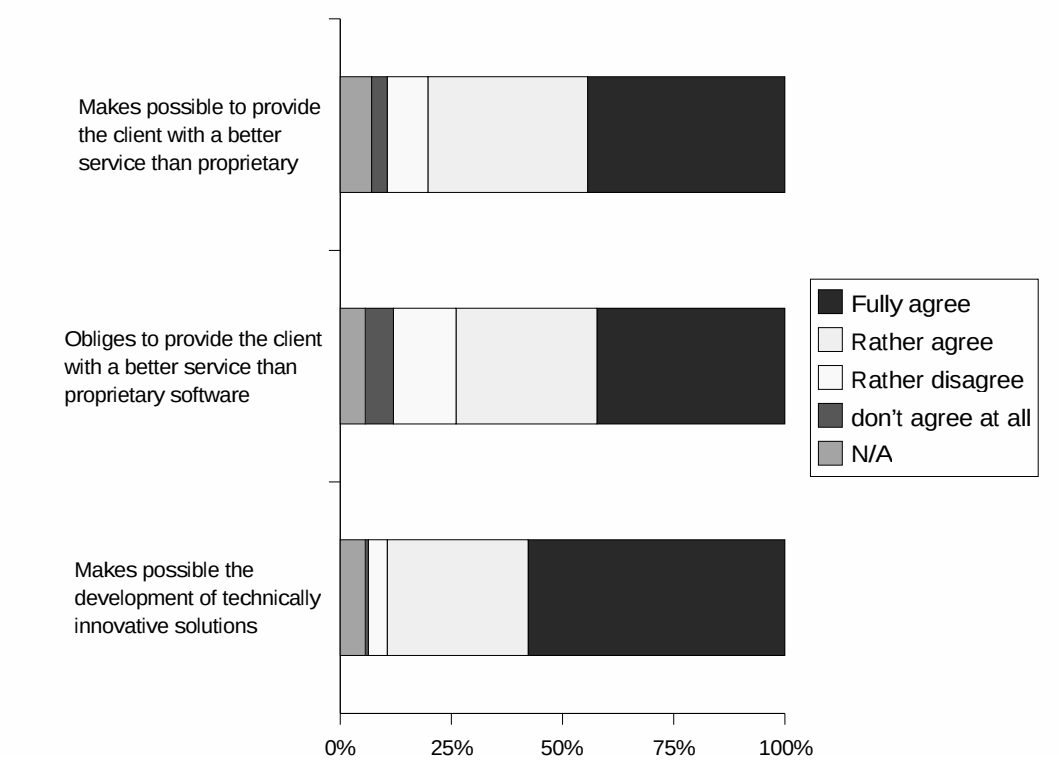
We have already said that the most mature free software programs were of very high quality. This facilitates the relationships between the producers of utilization technologies and those who use these technologies. Producers can more easily guarantee, through a contract, the reliability of the free software programs they use, because they are able to evaluate their quality thanks to the norm they have set up during the development phase. An assistance network is available to them and they can also intervene themselves on these software programs. In addition, the fact that the software program sources are accessible and that the evolution of these programs is not controlled by a firm, can reassure the adopter: the solution respects and will continue to respect the standards. It will thus remain inter-operable with the other programs he/she uses. The fact that firms use free software program services can be seen to be the creation of a professional standard since, collectively, their goal is to coordinate themselves in order to choose components and software program bricks which are both reliable and, especially, "standardized"[15]. Right now, this is lacking within the information technology industry (Dréan [1996]).

[13] It is the Free Standard Group (http://www.freestandards.org/), that brings together the Linux Standard Base (LSB, www.linuxbase.org) and the Linux Internationalization Initiative (LI18NUX, www.li18nux.org). The following are, among others, members of these committees: Corel, RedHat, Mandrake, SuSe, VA Linux Systems, Turbo Linux, but also IBM, SUN, SAP, Caldera, SGI, or the Open Group (in charge of the standardization under Unix).

[14] This is called the "TCO", for "Total Cost of Ownership". Today, Microsoft defends the idea that, if its software programs are more expensive than free software programs, they have a lower TCO, because it is easier to find firms that install them, given that their evolution is guaranteed, controlled by a firm, etc.

[15] In the sense that they respect public formats whose evolution is decided collectively.

Figure 1.2 — free software programs in service relationships.



The pooling of software bricks is also changing the competition among service firms towards long-term relationships and maintenance of software programs. It has become more difficult for them to pretend that the malfunctioning of a software program they have installed and parameterized is due to a program error. This can encourage firms to improve customer services and allows us to say that, in this field, free solutions can be competitive. The fact remains that firms must be able to benefit from their activity and, therefore, control part of the added value.

This is undoubtedly the most delicate point to defend today. There are few examples of profitable firms and many, still, have not reached a balance[16]. However, we have retained the following points:

- With regard to production costs, thanks to construction module, the cost for developing software programs is more broadly spread over time, thus resembling a service production structure whereby the missing functionality is developed only when necessary. The contribution of the service firms does not relate to the entire production of a software program but to the production of these components for clients who prefer free software programs so as not to depend on their supplier. Moreover, a component that has been developed for one client can be re-used to meet the needs of another client. A "security deficit" that has been detected for one client can be corrected for all the clients of the firm. As a consequence, firms monopolize part of the economies of scale generated by the collective use of a software program. In exchange, they guarantee the diffusion of their innovations and corrections[17]. One may say that service firms which base their offers on free software programs propose free "codified" knowledge, that is, software programs in order to sell the "tacit" knowledge they possess: the way software programs intimately function, the capabilities of their developers to produce contributions that work, to have those who control the evolution of software programs accept these contributions, etc. These firms are the most

[16] In the first group, one can name ACT Europe, Easter Eggs.

[17] This is one of the software editors' traditional roles. They finance this activity by producing and selling new versions of the software program.

competent to take advantage of the benefits linked to the apprenticeship generated by the development and improvement of software programs.

- because of these effects of apprenticeship and because it is difficult to diffuse the tacit knowledge one needs to master in order to follow and influence the evolution of a free software program, this role will inevitably be limited to a small number of firms. They will bring together specialists in software programs and will place them at the disposal of client-firms. They will have created strong trademarks, recognized by the user- developers of software programs and known by other clients. This will make it possible to diminish the pressure of competition, thus ensuring their profit margins.

Such competition also encourages these producers to contribute to the development of the software programs they use. First of all, it is a way to make themselves known and show their competence as developers to their clients.

Because every client has different needs, it is important for these firms to master a vast portfolio of software programs as well as to contribute to the development of standard software programs which are used in most offers. They must be able to present their clients with realizations that are linked to their problems. Finally, it is easier to follow the evolution of these software programs if one takes part in the innovation process and it makes it easier to understand other people innovations (Cohen and Levinthal [1989]).

This explains why these firms devote part of their investments to the development of free software programs, without these developments being directly useful (i.e. applicable to a contract). This is a research activity and this is what makes them different from "traditional" service firms which often devote less than 1 % of their sales turnover to research (see Genthon [2001]).

In a market based on the valorization of technical expertise, this contribution activity reinforces the image of a firm with regard to its expertise and capacities to be reactive, two qualities which allow it to highlight a special offer as well as to improve its reputation (via the trademark) and increase margins. On the other hand, this once again reinforces the tendency to concentrate on specific activities because it is necessary to lower research costs and, therefore, to increase the number of projects and clients.

There would thus be a system which, thanks to its openness, would facilitate service relations, allow the production of software programs while guaranteeing long-term incomes for suppliers. Obviously, just as the preceding developments did not eliminate old organizations, but rather reduced their field of application, in the same way, free software is not a universal solution. According to the types of software programs and types of use, the present organization can still remain the most effective one. One can even imagine hybrid solutions, similar to the ones SUN or Tech Troll propose with semi-free license systems or double licenses (Cf. Laure Muselli's article). We will thus finish this part of the introduction by specifying the criteria which are favorable to the publication of a software program under a free license.

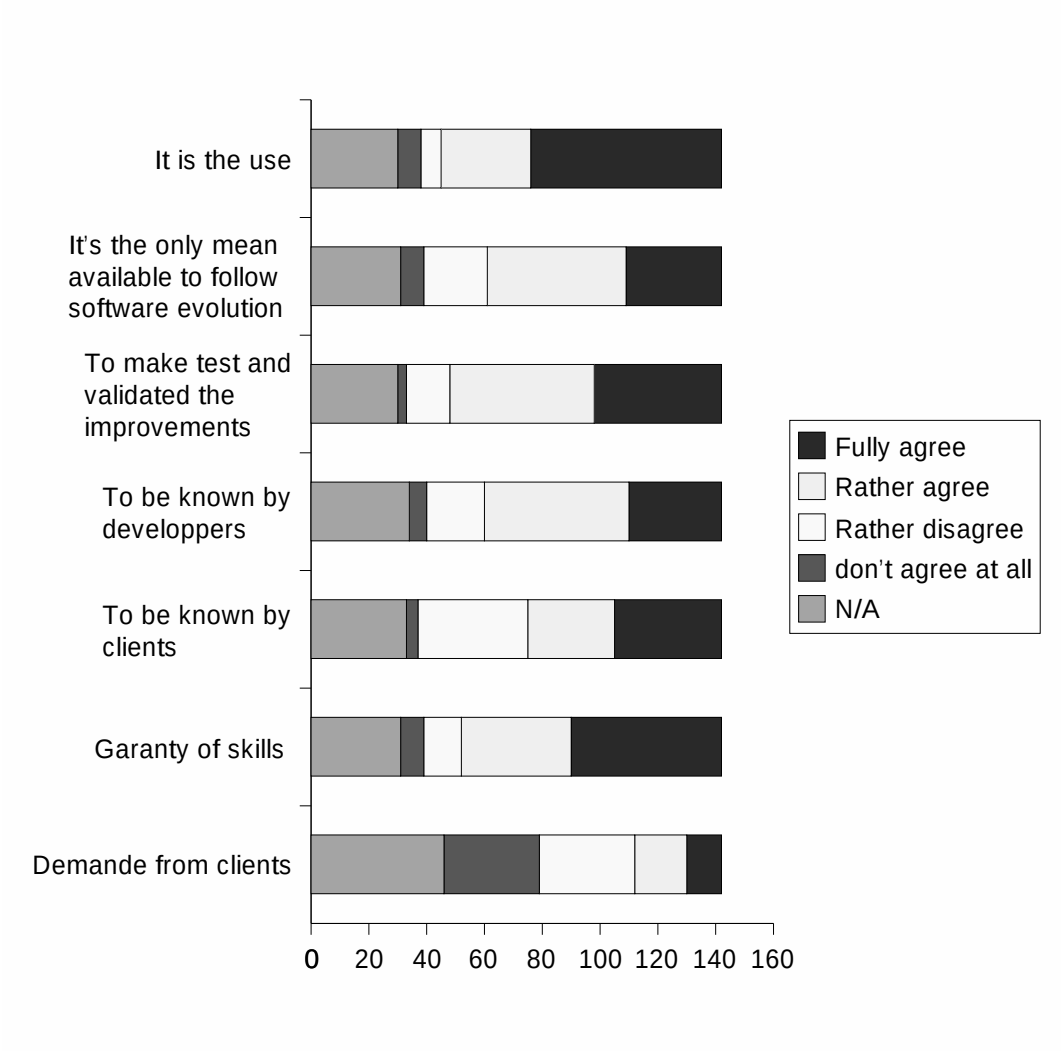
What are the markets for the free software industry?

Network software programs are the prototypes of the kind of software program for which a free publication is effective: these are software programs used by computer scientists. They require strong standardization and evolve quickly.

The existence of advanced users makes it possible to hope they will make contributions to the product which will reduce development costs, eliminate bugs or create new functionalities. Even if this pool of specialists exists, they still have to take the step to participate in an active way.

The fact that the software program is networked and standardized is a strong argument in favor of its being free, so that norms will definitely be respected. It also energizes the dynamics of innovation : from the standpoint of user-producers, the faster software programs evolve, the better it is to divulge the improvements that are made because such a dynamic is likely to make them obsolete (if the same functionalities are developed

Figure 1.3 — The reasons why it is necessary to contribute to free projects.

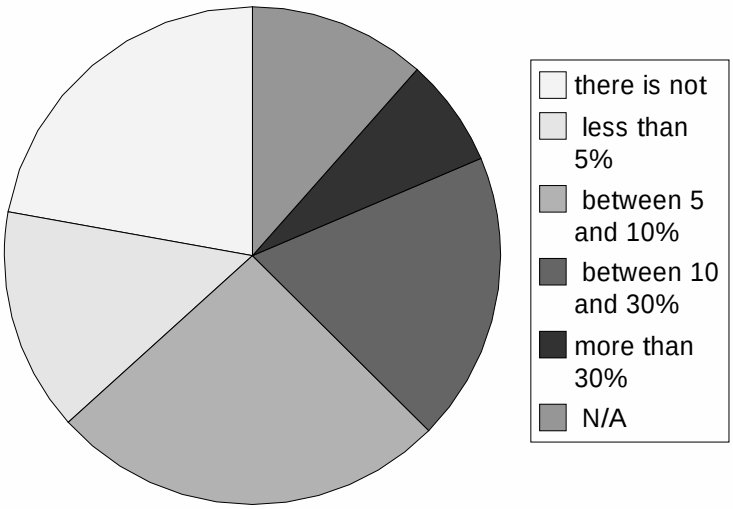


by others) or incompatible with the new versions of software programs. And this increases maintenance costs. Yet, the more the newly developed software programs become significant in the strategy of a firm, the more it will tend to keep them secret. These users-producers will contribute primarily to the development of software programs that evolve quickly. They are usually not very specific to a profession (the contributions which are made do not bring any information on the technological or industrial choices of the person who created them), or are very heterogeneous (the contributions will not be easily usable outside of the organization which proposed them).

As far as producers are concerned, these same software programs bring the biggest opportunities in the field of service development (followed by software programs, the creation of new functionalities and their integration in the official version, etc.). Users can also evaluate the technical quality of the developments proposed by the producers and propose their own contributions. This is particularly true with a dominant, proprietary type of offer. Being diffused under free software programs makes it easier for users to test this alternative.

On the basis of this analysis, one can attempt to decipher not only the position of free software program producers or commercial businesses but also individual users or research centers.

Figure 1.4 — Free time allotted to develop free personal projects.



1.2 Producing and developing free software programs.

We will now focus on the motivations of organizations which produce free software programs since individual motivations have already been the subject of several analyses[18]. There are three main types of organizations: companies which produce in-house software programs, research centers, especially public research centers and firms that put together commercial offers.

1.2.1 Developing standard software programs and valorizing in-house software programs.

Like firms, users are "in competition", that is to say that their work can also benefit their competitors (like for software program producers) and, as we explained in the preceding paragraph, their contributions will deal mainly with technical software programs, not particularly "trade"-oriented and which evolve quickly. Indeed, they are the ones that minimize the share of competitive advantage due to disclosure.

This strategy is well illustrated through the Code-Aster" case. EDF developed this software program to model structures and constructions. It is not part of EDF's job, but it is rather significant for the French company (it is used to study the aging of nuclear thermal power stations), therefore it is necessary to improve its safety. As François Horn shows in his article, choosing free software, when software programs are produced within firms from which this is a secondary activity, meets two principal objectives. The first is the diffusion of software programs so as to increase the user base, thereby improving the conditions of utilization in order to increase its reliability. The second objective is the redirection, if possible, of part of the maintenance costs towards these new users.

However, this must be regarded as a strategic choice that implies investments. Indeed, releasing a code requires that it be readable. This often involves rewriting it and getting more documentation. Guillaume Rousseau estimates that this amounts to approximately 20 % of the development cost[19]. As Alain Conchon (Alcatel) pointed out, "one does not want to give a bad image of the firm either by publishing badly written

[18] Among others, Lerner and Tirole [2002], Lakhani and von Hippel [2000], Foray and Zimmermann [2001], Bessen [2002], Jullien [2001].

[19] Here, we are quoting talks that were given the second day of this seminar. They are available on http://wwww-eco.enst-bretagne.fr/Etudes_projets/RNTL/workshop2/.

codes or codes that cannot be read properly"[20]. Then, this kind of publication sometimes goes against the culture of firms which is "not always oriented towards co-developing software programs or even publishing in-house source codes" (Gérard Giraudon, INRIA), "even when it simply is about "pseudo-free" programs." This means that developers must give up their code so that it can be adapted by other in-house developers" (Alain Conchon). Lastly, diffusing software programs and creating a community of specialists are not things that can be done just like that. It is often necessary to make a considerable effort to facilitate the access to and handling of these software programs in order to make them compatible with the tools that are used upstream or downstream from the chain of production in which they are integrated.

To conclude, let us stress how important these contributions are for the production of innovations in software industry. These users are, indeed, "early users", that is to say, entities who express needs which might become those of the market several months, even several years later and which strongly valorize the creation of a solution to such needs (von Hippel [1986]: 796). Therefore, these are users who are ready to invest in order to develop solutions: we are thinking here of Ada 95[21], but it also corresponds to the EDF's situation.

The type of license that is chosen for the use of the software program proves to be highly significant for the following two reasons:

- if these firms seek to diffuse knowledge and develop initiatives involving their software program, they do not want to be dispossessed of their work either and become dependent on a producer who would have taken the code. Worse still, they do not want to lose control of its evolution;
- the diffusion of a software program involves legal responsibilities, in particular on the consequence that the faulty design of a software program. A license can help deal with these responsibilities.

Finally, even if the relationship with the users is not primarily based on financial valorization, it clearly costs a lot of money. The license contract proposed by the producer can help in a more or less effective way. We will find this type of situation in the case of public research.

1.2.2 Valorizing software programs which stem from public research.

Undoubtedly the first producers of software programs in France are public researchers. They also wonder about their relationship with free software producers.

As Guillaume Rousseau —basing himself mainly on his analysis of the way valorization at the University René Descartes is dealt with— and Laurent Kott (INRIA valorization manager) both underlined it, publishing under a free license suits the customs and habits of research rather well. Indeed, its main reward is one's recognition by one's peers. This requires their judgment and the publication of one's work whose essence is today, and in a great number of disciplines, presented in the form of algorithms.

On the other hand, these institutions aim at valorizing the work of their researchers. When it deals with computer programs, the institutions are the proprietors and, as such, they can be held responsible for any possible dysfunction. It costs them money to publish free software programs. Thus, "the effort INRIA is making today in the promotion of free software programs represents the work of 20 men per year, to which it is necessary to add the efforts involving the creation of consortia: around CAML, Scilab, etc." (Laurent Kott).

In spite of these difficulties, it seems that research organizations want to develop their free software program portfolios. For was for pragmatic reasons initially. Knowing that it is difficult to prevent researchers from

[20] Besides, François Horn points out that during the first work meeting, a representative of Thales had confirmed that "[they had] some software programs [they were not using] anymore, but [they did not dare] show them because they had not invested enough on the documentation and on the structuring of the code".

[21] The American Department of Defense financed this language. It wanted a robust, object programming language whose evolution would not be dependent on the strategy of a firm (that is why it is published under a GPL license).

publishing their software programs, it is undoubtedly more effective to accompany them in their procedures, to offer them solutions in terms of licenses and support (servers) for publication which satisfy them. First of all, this allows them to enhance the reputations of their institution, not only among research centers, but also among firms which can more easily test these software programs. A policy of expansion as well as a transfer of technology can be conducted which creates employment in the private sector and financing for public research. The example of "Alliance", the software program library that Guillaume Rousseau presented in the second workshop, illustrates this very well. To this day, half of the doctoral students who work within the "Alliance" innovator laboratory are financed thanks to the valorization of these software programs. Two service and software program development firms which produce and distribute components of the library under proprietary format have been created.

It is possible to valorize public research thanks to free software programs. However, this requires a strategy, imposes some constraints in term of the quality of the software programs that are provided. In fact, these are the same constraints software program user-developers have to deal with. Industrial standards must also be taken into account during the designing phase, service support businesses must be developed if necessary and, of course, in this case, a well-adapted license policy should be elaborated too[22].

This policy will have to take into account the characteristics of both the software programs and their market. These characteristics determine the type of commercial activities which can be developed based on these free software programs.

1.2.3 Inventing free software program markets.

Before we study how free software programs allow users to invent markets in the field of information technology or, at least, to make such markets evolve, it should be reminded that quite a few contributors, like IBM, SUN or Netscape, make offers based on free software programs and contribute to their production for strategic reasons which do not necessarily have to do with the free aspect of the programs they support (even if one finds arguments such as those put forth in free licenses, the creation of open standards, of "norms").

At the same time, these strategies facilitate, and sometimes even allow, the diffusion of this model. The story of the PC illustrates this point.

Strategies of traditional producers: the example of information technology builders.

Dréan [1996], p. 230) notes that the two competitive driving forces in the field of software packages are "the competition between the leader of each sub-sector and its challengers" and "the on-going expansion of the information technology application field, which is opening up new domains [...] in which, on condition that he/she is the first to come in, every new candidate can hope to become a leader".

Therefore, when one wants to impose a software program, a language that will become a standard, it might be wise to take a close look into the software programs which use it or cooperate with it. The best example of this type of behavior is undoubtedly the work that SUN has done to develop the Apache free software program. This firm deals with everything that relates to the adaptation of Java to Apache. By rendering the language it has developed compatible with the market standard, it can hope to sell the Java development tools

[22] Concerning this last point, in France public organizations **must** write their contracts in French. This makes the choice for a GPL license more difficult. Participants, along with Mélanie Clément-Fontaine, have remarked that "the law is much more demanding for legal entities under public law or in charge of a public service". They also encounter difficulties in "the transcription of GPL licenses in both the French language and French law which do pose some real problems, particularly, with regard to the terminology". Participants have insisted on the importance of the efforts which should be devoted to the translation of these licenses. They would like to have a license model which would be common to all public organizations. This model would make it possible to develop valorization policies on solid, legal bases and to create a critical production mass which would guarantee its recognition within industrial circles and in the eyes of developers.

which it produces (as well as its expertise in this field).

Producers are willing to finance a platform in order to support software programs which function thanks to them. These can be viewed as crossed subsidies whereby an open standard tool comes as a complement to the company's software program offer. This is traditionally done in information technology: Genthon [2000] stresses that, to support one's standard (or product), it is necessary, among other things, "to establish alliances with co-producers" and "to sponsor the first users and the first complementary products".

More generally, challengers may want to subsidize a standard which is in competition with the dominating standard. The support computer builders have shown Linux definitely has led to the success of Microsoft and its dominant position on the PC market. Moreover, it is in the interest of businesses which depend on a standard to carry out their activity (like SUN with Apache or PC manufacturers and software programs that come with the operating system) that this standard be as open as possible, so as not to depend on the strategy of the firm which supplies it. This allows them to remain in charge of its evolution or, at least, make sure that this evolution does not work against them.

This standardization process is a typical example of the kind of situation in which each participant wants to promote his/her own solution, but stresses the existence of a standard more than the success of his/her proposal itself. Let us develop this point which is of utmost importance for the diffusion of free software. Today, traditional manufacturers are reproducing the same type of behavior that they had in the past vis-à-vis innovations such as Unix or micro-computers. Aware that there is a demand, they seek to integrate the new free offer in their own offer portfolios, as they did with the preceding innovations (Unix, micro-computers, etc.). This is logical since they do not control a dominant standard. In this case, it is in their own interest to support the development of a competing standard. By so doing, they legitimate the free software offer by placing it at the same level as the other operating system offers and thus facilitate its diffusion. This attitude is illustrated in our report through the interview of the IBM Linux manager in French-speaking Europe.

The first impact will undoubtedly be for the market of computers under Unix. Unix is the second market in terms of value after that of micro-computers. However, it is also a more divided market. In its organization, it resembles the traditional information technology industry more than that of PCs. As a consequence, as Genthon [1995] shows, it is less effective with regard to the production of hardware than that of PCs, so much so that it is now possible to compare their material performances. Unix computer manufacturers are competing today with workstation manufacturers with an "Intel" type of architecture. The latter mainly use Windows 2000 operating systems (at least on small server markets). The arrival of free software programs on this market has had a double impact: on one hand, it creates a standard Unix offer which is independent from the platforms, and it reinforces the attractiveness of the Unix systems (which provide the privileged support of free-use software programs, even if most of them also work under Windows). On the other hand, it puts more pressure on computer manufacturers under Unix by leading to the standardization of the operating system offer, and because GNU/Linux makes it possible to propose a Unix system that works on PCs. Basically, and as far as hardware is concerned, this diffusion only reinforces an on-going process: the diffusion of the open structure of the PC industry to the whole of the industry (this is what Horn [2000b] called "the underlying fusion of networks"[23].

Other traditional producers are witnessing the evolution of their businesses, though without it being really called into question: those of component manufacturers and editors, software tools such as data base software programs or compilers.

The manufacturer-editors of technical software.

In the field of technical software programs, whenever there exists a free software program, it is often under the control of a firm. The latter may be at the origin of the program (like Digital Creation with Zope or Matra

[23] Also see Marie Coris' article and Jullien [1999].

Datavision with Open Cascade) or may have been created by people who are part of the core of developers (like ACT with Ada 95, Scriptics with TCL, Cygnus with GCC, etc.). These firms control and guarantee the evolution of the software program and sell service assistance along with their tools. For these firms, the trademark is linked to the software program trademark and they often seek to be recognized as the firm that discovered the program.

What follows explains the importance of the "free" software strategy. These technical software programs constitute the basis for any information technology infrastructure. Therefore, these software programs must be perfectly mastered by their users. They must be able to configure them, guarantee the dependability of the IT installations and adapt the services to the needs of these users or organizations which utilize them. In this context, the demand in terms of quality, insurance-quality, respect of the standards is highly relevant. Their users particularly appreciate the opening of the source-codes and their non-appropriability, especially since they also are the best informed of the existence of free software programs, the best capable of evaluating them and adapting them to their needs.

It is also in these domains that the use of free software programs is the oldest, in particular thanks to internet. Innovations that come from both researchers and users in general are the most numerous and are favorable to the diffusion of free software programs. Finally, it is often the case that producers find it more beneficial to choose a free type of strategy for these software programs: markets are service markets (these are tools that must be adapted to the specific needs of the relevant organizations). Users are capable of evaluating the technical quality of the developments proposed by producers and of proposing their own contributions. This is especially true when there is a proprietary type of offer which is dominant: the diffusion under the free mode makes it easier for users to test this alternative.

Nevertheless, the business remains rather close to that of proprietary software-tool builders, like Oracle or Ilog. All these firms sell support, generally under a fixed price. They also sell component adaptation to final users or to other utilization technology producers. More than a revolution, the evolution towards free software accompanies the evolution towards the production of components. This increases the need for high quality and especially for the guarantee of standardized interfaces: in short, for services of "maintained, available technical capabilities", to quote Gadray's classification Gadray [1998].

In practice, these "component producers" must take up a commercial challenge, a challenge that is stressed by Pierre Bruno, the general manager of Open Cascade[24]. Service offers must be clearly defined in order to transform a significant part of these software program users into potential clients. The specificity of these firms, compared to builders, software publishers or service firms, which we will talk about in the following paragraphs, is that they suffer most of the maintenance costs and especially those of free software program development, on which they base their offers. Therefore, the objective is to transform a disadvantage (significant investments) into a commercial advantage, by either proposing certified "official" versions[25] or by establishing double license strategies, that is to say, by selling the most recent version under a proprietary license and by freeing the old version in order to make the product known.

Nevertheless, a firm like ACT Europe, which controls the Ada 95 GNAT compiler, and for which Aymeric Poulain-Maubant has proposed a monograph, still remains profitable because it succeeded in selling the certification of its software program and, more particularly, development assistance. An important factor is that it did not have to bear the original development costs of its software program. In defense of this firm (or rather of its economic model), it should be noted that it financed this development (and protection through the GPL) because it estimated that this was the best system to guarantee a quality product as well as its independence with respect to its producer. In addition, it is quite common for public researchers to develop proprietary software programs[26]. It is thus not surprising that strategies which supposedly make services bear the entire

[24] François Horn introduces this firm in his talk.

[25] Which means guarantees on stability, response time, when a problem occurs, etc.

[26] To show an example: Internet, or, more recently, Unix BSD, taken by APPLE with MacOS X and by Microsoft to build certain elements of the Windows programs.

development costs and which are more expensive, as illustrated through the Open Cascade example (developed in François Horn's article), have less conclusive returns on investments, because they are spread over a longer period of time (but it also the case with a license system).

But even in this most unfavorable case, we must admit that this economic model seems more coherent and more solid than that of the editor model. This is particularly true with regard to Linux distribution editors: their competitive advantage and share of added value seem threatened, both by IT manufacturers and by service companies.

Distribution editors.

Mandrakesoft, SuSE or RedHat differ from the preceding firms in so far as they specialize in the assembling of multiple software programs around one operating system. They position themselves between computer manufacturers and functional software program manufacturers. What seems fundamental to the GNU/Linux diffusion and, thus, to the free software phenomenon, is that distribution producers always seem to look for a perennial model.

The example of Mandrakesoft in France, about which Aymeric Poulain-Maubant gives us more detail, echoes that of RedHat in the United States or Suse in Germany. It shows that, today, it is not profitable to produce and distribute standard GNU/Linux distributions through the usual networks (bookstores, IT retailers). The limits of such a positioning lie in the fact that, since these distributors use free products, they can encounter competition since similar products can distributed by other producers. Free software does not constitute a source of income but is simply a loss leader for maintenance services. As is the case for traditional editors like Microsoft, developing a distribution trademark is of utmost importance since it demonstrates the quality of the products. It enables them to reach a segment of the public, that of "naïve" users or at least non-experts. The trademark is the only thing the latter have to evaluate the quality of the offers aside from the information that the price of the product can give them. Once a significant segment of public is seduced, they hope to develop resources by proposing services with added value, thus increasing their "ARPU"[27]. Because of the effects of standardization, these offers will probably be limited to some main distributors, even if they may gradually become available through several local editors/distributors.

Two economic strategies have emerged: they either specialize their distributions (RedHat and, to a lesser extent, Mandrakesoft), or they develop standard service offers, which are more public-oriented (Mandrakesoft has come up with the idea of users' clubs, whereby, in return for a yearly subscription, they have access to various on-line services and software programs). The second strategy consists in standardizing, industrializing the trade of architecture builder. This includes strong guarantees, after-sales warrantees through standard assembling of software programs and standard assistance services.

The fact that services are becoming more and more customized puts the editors in competition with service firms. They strongly base themselves on the way component manufacturers function. This implicitly shows that there is a competitive advantage in developing a distribution, in term of controlled levels of competence. In this case, it is not certain that proposing a succession of software programs is an advantage. Indeed, clients may be afraid to be dependent on the distribution editors for the coming developments. The risk is there that, in order to increase their margins, these producers will propose improvements which are not totally compatible, so as to differentiate their offer, but also to make their clients more dependent. However, the freeing of software programs should guarantee sufficient compatibility between the different versions. The competition would be more similar to that which exists between producers of compatible goods and would allow several offers to survive (especially since the principal "editors" get involved in standardization bodies such as the "Free Standard Group" http://www.freestandards.org/news.php?id=35).

[27] "Average Revenue Per User". This term is mainly used in telecommunications and makes it possible to evaluate the profitability of a firm by basing oneself on the average income generated by a user.

For the time being, the move towards users' clubs has not helped Mandrakesoft to make profits, even if the fact that the initiative is less than six months old certainly does not allow us to conclude on its relevance.

Finally, bringing together SuSE and IBM, or Mandrakesoft and Sun (via the diffusion of Star Office), shows that IT builders are starting to invest this field. It is not foreign to them since they distribute Unix versions in order to develop the service component of their incomes. This is likely to reduce whatever room is left for independent editors.

Considering European service traditions and considering the characteristics of the free software industry, it is not surprising to note that most firms choose to concentrate on services based on free software programs[28]. Do they propose a new model, what is their specificity with respect to traditional service companies to SSIIs ?

The SSIIs: Service and computer engineering companies.

The latter position themselves on expertise markets, their goal being to develop individualized solutions and to maintain these solutions, instead of having the client do this. This is close to what Gadray [1998] calls "providing human capacities", in the sense that what characterizes these firms is that they regroup a team of specialists in different software programs. User-developers or designers make their reputation in the same way they build up their own expertise; by contributing to many different software programs. Free software programs are used because they allow for the creation of more flexible solutions which are not necessarily attached to either editors or manufacturers. In the long run, they also provide more balanced service relations[29].

They must integrate two contradictory requirements. First, it is necessary to understand requirements which are linked to the profession, which means to be able to base oneself on functionality libraries, and therefore on software program libraries. Secondly, one must simultaneously be an expert on certain software programs, this expertise being the only factor that can create added value to the use of free software programs.

There are three ways of solving this dilemma: either free software companies concentrate their energies in order to combine their expertise and plurality under the same structure, or they manage to regroup specialists in software program technologies. The third choice is to become subcontractors for traditional SSIIs.

This last possibility is obviously the least favorable one for SSLLs (companies specialized in free software services)[30]. They are likely to be hierarchically integrated, or at least find themselves in a position of dependence. This situation would be similar to what happened to Bouygues where the IT department contracted out to a myriad of more or less specialized small companies.

But if the first two alternatives offer broader prospects, they are not without problems either. In the first case, there is a risk of losing what seems to be one of unique characteristics of the SSLLs : the management of human resources. SSLL employees benefit from a considerable autonomy in their work. It is considered t be something normal to give them extra time so that they can contribute to free projects. Recruiting through co-optation is very common when people who work on the same projects are used to working together[31]. If these firms merge and, as a consequence, become larger, it might be more difficult to handle this "family" type of management. Indeed, collaboration is often informal and the line between work and one's personal activities can be quite thin. Lastly, these firms are likely to run up against distribution editors" and even against computer manufacturers whose business seems to be increasingly oriented towards the assembling of software

[28] Marie Coris' presentation deals with these firms.

[29] One of the conclusions that comes out of the data processing in our investigation is that one of the characteristics of the SSLLs is that they favor GNU/Linux Debian distributions to build their service offers. It is the only major distribution which is not controlled by a firm.

[30] At the beginning, the term "SSLLs" was rather consensual (during the first meeting, these companies claimed it was a distinctive sign). According to participants, today, it seems to have lost its full meaning, undoubtedly because one of the companies has made it a registered trademark. We have retained it anyway, because it is commonly used today, in particular in the IT media.

[31] Certain companies, like Easter Eggs (http://www.easter-eggs.com/), even propose particular industrial structures, whereby the company is equally owned by the different employees.

programs in a broad and varied distribution that corresponds to the needs of the clients.

The second case obviously deals with coordination problems software specialists often encounter as well as the identification of functional skills required for software program solutions and the management of these solutions. Finally, specializing in one software program is effective only if it generates an added value and if one knows how to sell it. The example of "component producers", like Open Cascade or ACT, shows that this model has not yet been stabilized either.

Choosing the right economic landscape.

Aside from this classification, this analysis shows that the various strategies are a hybrid and there is still no single economic model that is the most efficient or most profitable. It is quite clear that it is extremely difficult to propose a prospective analysis of the evolution of the economic landscape because there are so many factors to take into account. Yet, it seems that the very fact that the strategic behavior that has been noted above is persisting, makes it possible to describe two rather different types of landscape. As in the past, their choice will be conditioned by the strategic positioning of computer manufacturers and by the preferences shown by the users, and most particularly the industrial users.

Computer manufacturers must provide operating systems. In keeping with the "Unix" industrial logic, these manufacturers must control the development of the system whereas, following the approach of "PC" manufacturers, they have delegated this control to an external supplier. Right now, manufacturers tend to reproduce these same practices in the way they use free operating systems. When IBM or HP propose several distributions (HP goes even as far as choosing the Debian distribution as a reference), PC builders, like Dell, establish partnerships with distribution producers (RedHat for Dell). Even if these choices were foreseeable, it is still easier to break organizational and commercial "routines" than to make radical changes to them. They thus propose two types of industrial organizations.

If manufacturers regained part of their control over the operating systems, as in an industrial organization similar to that of a Unix-based offer, it would be to develop service offers. They would then be in a position to compete with service companies like Alcôve while free distribution editors would disappear. At the same time, using an operating system which they do not completely control always leaves them a chance to use one or more computer suppliers, along with a service company, which ensures their cohabitation. If it is the "PC culture" which is dominant, builders are likely to simply be component assemblers. In this case, the income derives from services being seized by the distribution manufacturers: RedHat, SuSE or Mandrake would then be the dominant firms of this new industrial period. The industrial organization would be similar to that currently found in the PC industry, where margins are extremely reduced for computer assemblers and very comfortable for the operating system and software program editors. But the freeing of the structure, which makes compatibility easier, should allow the survival of several suppliers. This would guarantee a certain amount of competition between the distributions. Operating systems would be more like components and would be installed by the manufacturers. Computers would thus perfectly match the clients' requests.

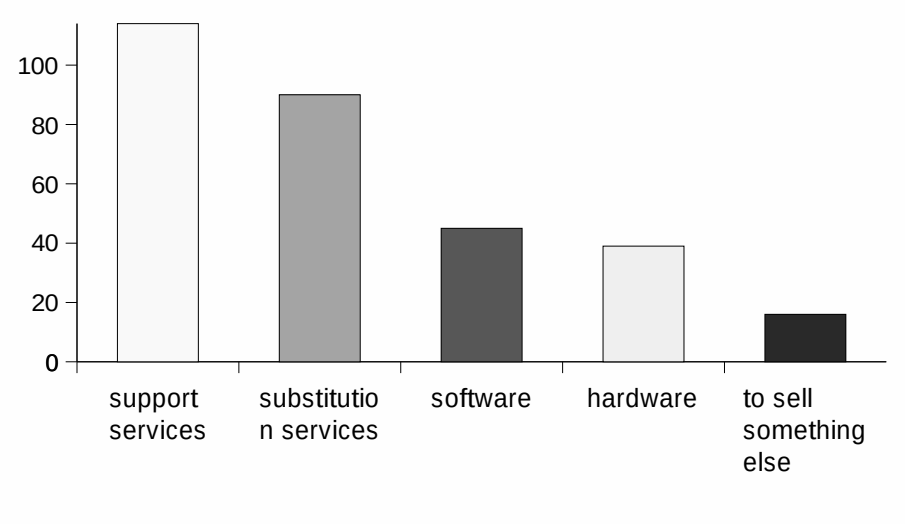
Computer manufacturers will be partly responsible for the direction industrial organization will take since they are now, thanks to their size, in a bargaining position with respect to distribution editors. However, this evolution will also depend on the user-firms since they are the ones which select the offers.

User-developers would probably favor the first system because it guarantees them the best control over the evolution of the software program they use, where the degree of standardization is highest and number of technical choices for the firm are the lowest. For these very reasons, this type of organization should benefit from companies which choose free offers because the latter guarantee the long-term survival of the software programs and their interoperability. But it is also the organization which requires the largest investments on behalf of the clients because they must either be able to follow the evolution of the standards on their own or ask a service firm to do this for them.

The second system is not as demanding for the clients as far as adapting their "routine" is concerned. It more similar to the PC world type of organization, with operating system supplier-editors whose trademarks are known. This system would undoubtedly be more readable for "naïve" users who would basically have to choose between various products rather than having to specify the way they want their computer to be built.

Depending on the users' objectives, one of these two systems will probably be the most appropriate. They may even cohabit: the first system would be used for the most technical applications, in particular for servers (for instance, during the first phase of the diffusion), the second system would be used when choices are less technical, when it is a matter of distributing the same system on a large number of stations, or when clients are less technically competent (small companies, for example). What seems quite clear, however, is that "there is only one way of making money: services".

Figure 1.5 — The number of firms which use free software programs according to the type of offers they provide.



Sometimes, services are too limited and the existence of a commercial activity might not always be possible. Once again, we return to that same question concerning "the link between commercial activities and free software programs" (François Horn).

This link is made through the contractual elaboration of the relationships, whether it be at the level of software program licenses (Laure Muselli analyzes what this means with regard to the strategic positioning of companies which use them), or at the level of service commitments. One clearly perceives the significance of the legal questions in the apprehension of free software ecology.

1.3 The future legal framework for the production of software programs.

Software programs have significant economic value and are part of the kind of immaterial creations which can be protected thanks to intellectual property laws.

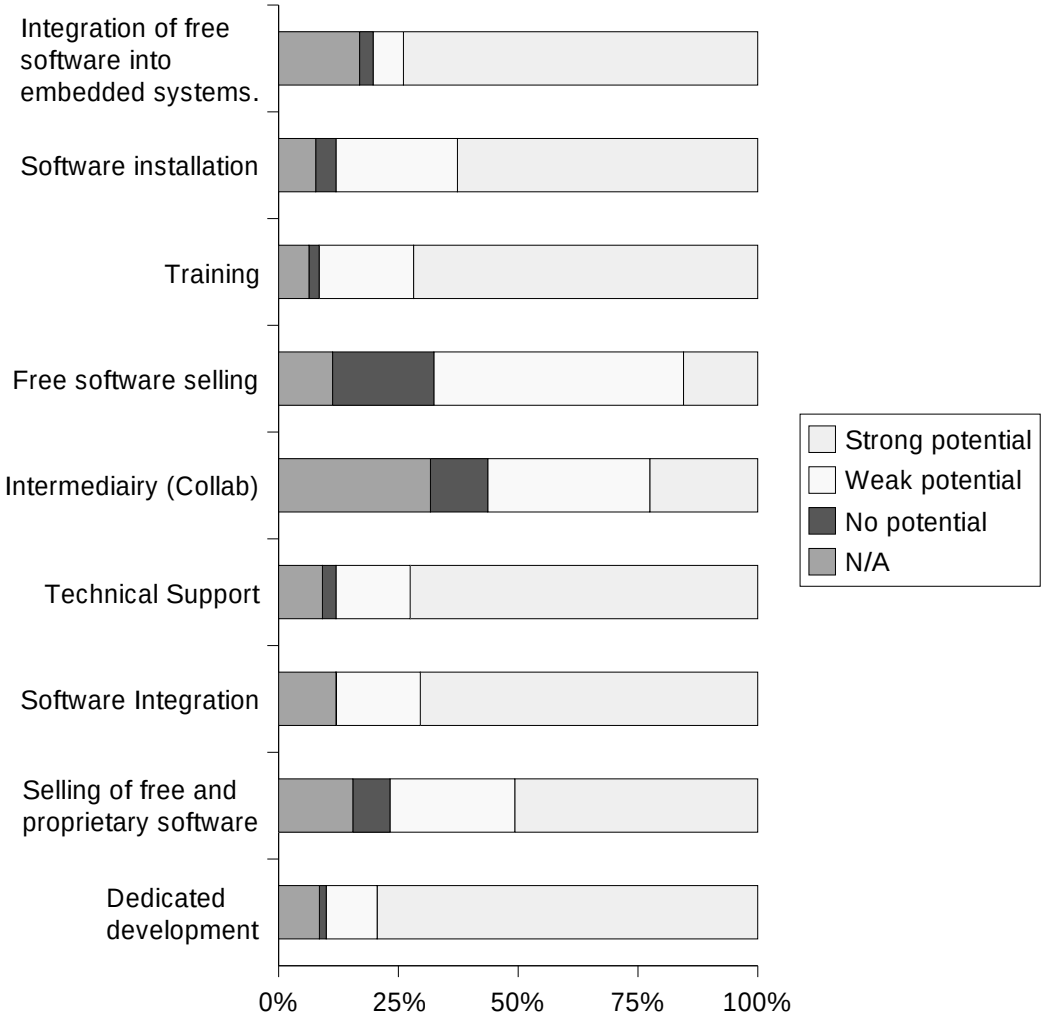
As early as 1968, French legislation[32] excluded software program patentability without indicating the type of regulation which could be applied. As a consequence, the courts were the first to consider **the copyright**[33] as a means of protecting software programs. The January 3,7 1985 law ratified this choice[34]. However, the

[32] L. n°68-1, January 2, 1968: textitJ.O. Jan. 3 p. 13. The same solution was adopted by the Munich Convention, October 5, 1973, article 52.

[33] See M. Vivant and C. Le Stanc and alii, *Lamy Droit de l'informatique* 2002, n° 151.

[34] Law n°85-600, July 3, 1985: *J.O. July 4, 1985, p. 7495.* In order to bring the regulations with regard to the protection of software

Figure 1.6 — Previsions made by companies concerning the potential for commercial activities which use free software.



technical aspects of software programs were taken into account, the result being that a rather large number of provisions which are applicable to software programs go against the authors' basic rights.

Among these rules is the automatic devolution of the patrimonial rights to the employer "for software programs and their documentation which were created by one or more employees in the exercise of their duties or according to the instructions of their employer" (article L113-9 of the Intellectual Property Code). Another example is the decision to make it illegal to produce private copies. This provision has been replaced by an authorization to make back up copies. The user may also benefit from rights which are more extensive than those envisaged by the common regulations. Thus, under certain circumstances, he/she can even reproduce, translate and adapt the program (article L 122-6 CPI). In addition, the protection of the integrity of the work does not have the same force anymore[35]. Indeed, software programs are considered to be a tool. As a conse-

programs into line with the different European countries, a Directive was adopted by the Community authorities on May 14, 1991 (*Dir. Cons. CE n°91/250, May 14, 1991, JOCE May 17, 1991, n°L 122, p. 42*. It was transposed to France via the May 10, 1994 law (*L 94-361, May 10, 1994, J.O. May 11, 1994)* .

[35] A. and H.-J. Lucas, *Traité de la propriété littéraire et artistique*, 2 ed. Litec, 2001, n°419.

quence, the legitimate user can correct or de-compile the code. Finally, withdrawal[36] and repentance[37] rights have not been retained (article L 121-7 CPI)[38].

The utilization that can be made of a software program is described in a contract which often comes under the form of **a license**, particularly when dealing with software packages[39]. If it is appropriate that the license respect the legislation which applies to the program, it still provides the person who delivers the license with the possibility of determining the conditions of the program's utilization. Indeed, legal provisions are often applicable in the absence of any contrary stipulation envisaged in the contract. For example, according to the L 122-6-1 article of the Intellectual Property Code, certain acts, such as reproducing, translating or adapting a software program when this is necessary for its utilization in accordance with its stated purpose, are not subjected to any authorization. However, the license can underscore the fact that the author, and the author alone, will be able to correct the errors and determine the specific modalities to which such acts will be subjected.

The person who delivers the license has enough freedom of action to define the authorizations he/she confers on the users. Free software program licenses illustrate this perfectly.

Before these licenses existed, there was a "free software approach" was not uncommon. Without it, Internet would definitely be different today. Indeed, the majority of internet bricks are free software programs: Sendmail, Postfix for e-mails, the BSD project for its major contributions with regard to the Net servers, Bind for the management of the names given to domains or Apache for the Web servers. Standards and protocols were placed at people's disposal who were asked to contribute to their evolution[40]. For example, this was the case the TCP-IP protocol. This took place in the early seventies when, at the time, there were some significant standardization organizations (ISO, ANSI, AFNOR, IEEE) and some powerful manufacturers who used to develop and push their own network systems (Digital, IBM, France Télécom...)[41]

The objective was to favor circulation and the exchange of knowledge and thus create standards that could be mastered by everybody. Most of these software programs are subjected to licenses which expressly provide users with the possibility of copying, modifying and diffusing the program (whether it has been modified or not). The first free software licenses emerged in the United States in the eighties on the initiative of research organizations[42] and private individuals[43] who developed their own software programs.

The need to formalize the identity of software creators under the form of licenses, a practice commonly shared among researchers, is a reaction to the fact that software markets tend to claim ownership for themselves.

Indeed, the commercial valorization of software programs used to be, and still is, based on a system of appropriation authorizing the utilization of a program. This was, and often is, done in exchange for payment : the person entitled to the rights is the only one allowed to operate the future versions of the program and does not permit it to be modified. If there happens to be a legal exception which authorizes the legitimate user[44]

[36] The "right to withdraw" allows the author "to withdraw his entire work from the market (i.e. to put an end to its commercial exploitation"). P. Sirinelli, *Propriété littéraire et artistique et droits voisins*, 1992 Mémentos Dalloz, p. 56.

[37] The "right to repent" allows the author "to alter his/her work". P. Sirinelli, *Propriété littéraire et artistique et droits voisins*, 1992 Mémentos Dalloz, p. 56.

[38] With regard to the regulations concerning the protection which can be applied to software programs, see: C. Colombet, *Propriété littéraire et artistique et droits voisins*, Précis Dalloz, 1999, p. 85 and f., P.-Y. Gautier, *Propriété littéraire et artistique*, 2001; Mr. Vivant (under the dir.), *les créations immatérielles et le droit*, Ellipses, 1997, p. 50; M. Vivant, C. Le Stanc and alii, *Lamy droit de l'informatique* 2002, n° 140 and f.

[39] Software packages are "standard" programs contrary to "specific" programs which are conceived for a specific user.

[40] The publication is then made under the form of a RFC (*Request For Comment*.

[41] See Yves Rougy's study, in this report, entitled "Free License Uses". It describes, in particular, the technical fields in which numerous software programs are diffused under free licenses.

[42] In particular, the University of Berkeley, California, or the X consortium of MIT.

[43] Thus, Richard Mr. Stallman is at the origin of the GNU project...

[44] The legitimate user is the one who has received the authorization to make use of the program as opposed to someone who uses a pirate version.

to modify the program in order to correct errors or to allow interoperability[45], this exception is made under certain conditions. Modifications are allowed only within the limits of the domain of the work to be done[46] and if the user has no other means of accomplishing it. The author can also decide to be the one to make the corrections. Lastly, these provisions should not be interpreted as a means to undermine the operation of the work. It is thus quite easy to take advantage of all these limitations to reduce the user's room for maneuver. He/she is then left without any control over his/her tool. Of course, he/she is not given the source code under such conditions. Thus, for the person who is entitled to the intellectual property rights, the strategy of these licenses consists in giving this person the possibility of intervening on the program code by preventing anyone else from having access to the source code and by forbidding any modification of the software package[47].

If average users are not aware of these restrictions, accomplished users regularly come up against these kinds of constraints. Therefore, the best way of responding to these "reserved" software licenses has been to create "free" software licenses in order to allow tools to be exchanged. This in fact happened and a "free community" emerged. It was mainly constituted of developers who, thanks to the freedom of action they now enjoyed, could fully use the software programs and even contribute to their improvement. Today, the movement has taken on an international dimension and extends far beyond its initial sphere. Software programs that were thus diffused entered the economic domain : more and more companies are users of such programs, in a cumulative way or not, or are editors[48] or free IT suppliers. Consequently, it is not possible to consider free software programs as a marginal initiative without any real economic impact. In this system, the software program does not constitute any real commercial value, since only services[49] are subjected to transactions.

These licenses were designed to take the opposite course adopted in American copyright law[50] and this move seems completely suited to French copyrights. Sometimes, the community and altruistic aspects of the free software movement have led some to wonder whether it was purely and simply challenging intellectual property rights. If so, free licenses could not be valid in France. Yet, if these licenses really do constitute a challenge to the commercial exploitation of intellectual propriety, they still respect the author's prerogatives. As far as moral rights are concerned, which are quite limited with regard to software programs[51], free licenses do not contravene the author's right of disclosure since one of the major characteristics of these licenses is to show the author's will to reveal his/her own creation. In the same way, free licenses do recognize the right of paternity since the development of the software is permitted. Sometimes, it is even clearly required to indicate the modification that was made so that it will not be wrongfully attributed to someone else. As for patrimonial rights, if the way they are exploited may come as a surprise to some, nothing forbids them. Indeed, article L 122-6 of the intellectual property law stipulates that the author has the right to authorize "the permanent or temporary reproduction of a program, whether it be the entire program or only parts of it, by any means, and under any form (...): its translation, adaptation, arrangement or any type of modification of the program and, as a consequence, its reproduction, its placement on the market, for free or not (...)". All things considered, the author can diffuse his/her program under a free license[52].

[45] According to the EEC Directive of May 14, 1991 concerning the legal protection of computer programs, interoperability means the "capability of exchanging information and to mutually use the information that has been exchanged"; quoted by M. Vivant and C. Le Stanc, *Lamy droit de l'informatique*, 2002, p. 1934.

[46] The term "work" is used here with the meaning of copyright: "the creation of the mind stamped with originality which, as such, gives rise to copyrights" (Gérard Cornu, vocabulaire juridique, association Henri Capitant, publisher: Puf, 1997).

[47] They generally go by the name of proprietary licenses so as to distinguish them from free licenses. In order to avoid the confusion which could lead the reader to believe that *only* proprietary software programs can be protected by copyrights, we will choose the expression "non-free licenses or "reserved" software licenses.

[48] Editors in the usual sense of the term, not that of the intellectual property law. That is to say, companies favor the development of free software programs.

[49] The services which are offered are, for the most part: technical assistance, engineering, but also warrantees and security offers...

[50] The expression "copyleft" was coined by Mr. Richard Stallman.

[51] Indeed, "except otherwise stipulated, the author cannot oppose him/herself to the modification of the program" (L121-7 CPI article). He/She does not have the right to repent nor to withdraw. He/She still has the right of paternity and that of disclosure.

[52] For a more detailed analysis of the clauses relating to the various free licenses, refer to our study entitled: "Free software program licenses, a legal study" in this report.

The legal landscape for software programs may change with the recognition of a solution which was previously dismissed, namely: **software program patentability**. Nevertheless, associations for the promotion of free software programs have warned against the perverse effects of such a change[53]. The question definitely concerns both economists[54] and lawyers[55].

Indeed, on one hand, it would be necessary to define the criteria that determines the protection of patentable inventions, in this case, software programs, or deal with the question of the accumulation of safeguards (patents and copyrights). On the other hand, it would be necessary to measure the impact that the recognition of the patentability of software programs can have on the delicate balance that exists among developers. The latter sometimes have very different profiles[56]. Therefore, even if free licenses for software programs may eventually be created, in practice, one wonders whether free software program developers will have a fighting chance when confronted with huge, powerful groups which are eager to maintain their control of the IT market.

For the time being, free software licenses must be examined from the standpoint of existing laws and the way they function. People must keep in mind that, just like the economy, the law and information technologies are subject to constant evolution.

[53] Example: a summary of the AFUL's position (July 2001) http://www.aful.org. Also see the petition against software patents and for the freedom of programming in Europe

[54] For example: Y. Dupuis and O Tardieu, *Les brevets superflus en matière de logiciel*, 2001 report, École des Mines.

[55] See the study proposed by Franck Macrez entitled "Patents concerning computer programs: the extent of the protection", in this report.

[56] Indeed, multinational companies, research laboratories or private individuals do not share much when it comes to developing software programs. Yet, they are subjected to the same laws. It thus becomes quite clear that the recognition of software program patentability does not represent the same thing at all in their eyes.

Part I

Studies on the evolution of computer industry: how do “free software companies” present themselves?

The manufacturers: the exemple of IBM.

AYMERIC POULAIN-MAUBANT,
Jipo Interactive, Aymeric.PoulainMaubant@jipo.com.

- person interviewed : Marc Joly, Linux Manager for France, Belgium and Luxembourg

- and reference site : http://ibm.com/linux/

2.1 A few reminders.

Within a little less than two years, and whereas the Free community was hoping for this – as it was hoping that big names in material and/or software computing would —, IBM built both an attractive strategy and Free offer.

In the middle of 2001, it was already possible for a few privileged people to connect themselves, via Internet, to IBM servers (the zSeries type which work under Linux), so as to test the various applicatives which had been placed on them. Once this had been completed, IBM announced the launching of several *mainframes* which would work exclusively under Linux. At the same time, IBM was launching an publicity campaign on TV and was the first to put the Tux penguin before the eyes of housewives (less than 50 years old). The campaign characteristic was to also mention the other operating and/or software systems. IBM's strategy vis-à-vis the Free community is to include in its message that Linux and free software make it possible to consolidate expensive investments through top quality servers. Yet, both large companies as well as small businesses are targeted, through the E-business policy activated by IBM.

IBM estimates that the market is there, especially with companies which are increasingly confronted with problems of major costs cuts. The various materials which are proposed (iSeries classes for small and medium-sized businesses, or zSeries classes for large-scale computing resources consumers) with Linux solutions are the equivalent of 15 to several hundred traditional servers. This should be enough to draw the attention of those who seek to equip themselves as well as those who seek to write off the cost of their investments.

To obtain these results, and support a volunteer, Free-software-oriented policy, IBM invested 1 billion dollars in 2001, mainly in R&D in the development of both its software and hardware under Linux. This investment made it possible for IBM to announce, early 2002, *mainframes* under Linux only, whereas in 2001, Linux was only an option. This shows how much IBM trusts these solutions. And if IBM mainly selected the eServer zSeries class to do this, it is because these servers were the only ones to show an above the average growth rate within this sector over five consecutive quarters.

If these declarations and strategy coincide with the companies request to consolidate their investments, they also correspond to the trend towards host computers. Indeed, to set up several servers which would be materially separated would be a rather expensive solution. Therefore, it seems that, right now, things are going rather well for both Linux and *Big Blue* free software ambitions.

As proof of IBM's determination, Marc Joly was appointed as early as January 2001 to be Linux Manager for France, Belgium and Luxembourg [1]. This is what he has to say.

[1] He graduated from Lille Business School and joined IBM after having worked in the banking sector for a while. In 1996, he became the "point-of-sale terminals" manager for France, Belgium and Luxembourg. In 1999, he was head of development for e-CRM solutions for the whole of Europe.

2.2 Strategy.

RNTL: *More and more companies are choosing Linux and free software programs as alternative solutions. IBM has made some major statements about this market during the last few months. It is reminding people that its customers' priorities are to consolidate their previous material investments and reduce their number of servers so as to improve daily management and reduce costs. What is IBM current strategy with regard to free software?*

MARC JOLY: It revolves mainly around Linux. Everyone agrees that the market is quickly investing in Linux and free software solutions, whereas, just two years ago, only a few enthusiastic pioneers and the first adopters who had chosen to do so, used it. In a relatively short time, Linux has gained its letters of nobility, in particular in Internet-oriented applications, or large *clusters* used by the scientific community. It then was used in critical sectors such as e-commerce or banking applications. IBM is thus backing up this movement by proposing the same level of support that normally comes with all our solutions. We believe we offer our customers, our partners as well as the Free software community innovative, high-tech solutions.

For Linux, IBM's offer is made up of three facets : servers, software, services. Our strategy can consequently be described as follows :

1. Create a line of high-tech products so that Linux and free applicatives can be used at their best. Then, add the necessary services to develop and extend these applications,
2. . Make sure that all our systems either, support Linux entirely, coexist with Linux, or are compatible with Linux,
3. Develop partnerships with the community of free software developers in order to constantly improve Linux.

RNTL: *Why Linux, and not FreeBSD or another operating system, or several systems for that matter?*

MARC JOLY: Linux is clearly the consequence of a consensus. It results from the simple observation of "market shares". Like any other company, IBM must survive. For customers, it is the market that provides meaning. In fact, Linux appeared as the operating system which made sense.

In addition, Linux is excellent for everything that has to do various material, various infrastructure support. This is also IBM 's strong point. We can actually say that *we are the leaders in terms of hardware infrastructures as well as software and service* . Both thus go together very well.

I could also answer the question "why Linux free software" ? To have the Linux source code allows the user to have a source code which is much better and much more stable than any other solution. Of course, the flexibility and freedom which are associated with free software make it possible not to limit oneself to only one operating system, not to be locked up in proprietary solutions and to save money, which is something proprietary software cannot do. We can thus:

- manage systems and data with applicatives which are perfectly adapted to people's needs and constraints,

- reduce costs and optimize fees on servers[2],
- offer the technical teams of our customers adapted, adaptable and comprehensible software environments.

RNTL: *Is this IBM Linux strategy original? Which added value do your clients see?*

MARC JOLY: We definitely did not create something that was entirely new. We simply associated Linux with our pre-existing IBM architectures. Besides, there is no way that Linux can replace operating systems already used by IBM. It should rather be considered as complementary. We want to realize the offer we have already proposed to our customers. More especially however, and this reflects the simplicity of our approach - we want to help them use the pre-existing infrastructure as well as possible.

Linux enables us to answer our customers' major concerns : reduce the ownership costs (TCO), obtain a better service rate and a quicker response time of their architecture.

If, for example, one takes the case of a distributed server company, it takes an average of between a few -hours and several days to add an additional server. This was the case until we presented our zSeries Linux offer [3]. It allows the user to open Linux virtual servers in only three minutes. New processors can be added in less than 24 hours. As a consequence, service and reactivity rates have been drastically improved. These flexible solutions answer our customers' needs as far as quickly increasing their business.

RNTL: *For the time being, you have only described solutions for either very large companies or small and medium-sized businesses. What about private individuals ?*

MARC JOLY: Right now, IBM's strategy, as regards free software, is only designed for server companies. This is meant to simplify and consolidate our customers' architectures. When the applicatives which are usually intended for the general public will be more broadly accepted within the Free community, maybe the general public market will evolve more clearly towards these solutions. But this is not the case right now. The market will be the decisive factor if we also strongly get involved in Free software for private individuals.

IBM keeps an eye on these evolutions. To come back to the business market, it is ready for all server or infrastructure-oriented applicatives , like proxies and DMZ. However, this is not the case for ERP or CRM type applicatives . In 2002, Web-oriented architectures, mail or print servers, applications which require high reliability and safety constraints are among the fields of predilection of Free Software. This does not really reflect the needs of the general public, even if it might have to work with Free Software solutions without being aware of it. Indeed, these solutions will proposed via on-line service providers.

IBM closely follows these evolutions via our technological development monitoring system. When applications which make it possible to use proprietary applicatives on free platforms will work [4], we will see.

There again, the market is the key determining factor. For example, if it means anything, WebSphere will be proposed with other free software bricks as it is currently proposed with Apache.

[2] http://www-1.ibm.com/partnerworld/pwhome.nsf/mktgsale/linux_workload.html.

[3] The other classes, iSeries and pSeries are also excellent.

[4] Lindows, for example (writer's note).

RNTL: *IBM is resolutely getting involved in Free Software and, at the same time, is announcing a record in terms of patents which have been registered.*

MARC JOLY: Yes, close to 3000 patents. Both are compatible. Both make it possible to protect and diffuse innovations.

2.3 Products and services offered.

RNTL: *You mentioned zSeries servers. Concretely, how does IBM Free offer function?*

MARC JOLY: It concerns servers, hardware, software, services and support [5]

Servers The xSeries, pSeries and iSeries classes [6] allow to work under a Linux environment, but it is with the zSeries class that one obtains the best quality in terms of service, flexibility and availability, as I mentioned earlier. Note that IBM does not tamper with the Linux kernel. In all cases, standard distributions (RedHat, SuSE, Caldera, Mandrake, TurboLinux) are installed.

Hardware With regard to NUMA-Q , customers who use *Linux Application Environment* above the ptx kernel, can work with either Linux or non-Linux applications. For AIX 5L , sources are compatible, there is a conformity with the emerging standards and a development environment which allows, very precisely, the development and deployment of Linux solutions on .
Aix THINKPAD follow suit. Some are provided with a Caldera distribution (in fact, Caldera is an IBM partner) to offer e-business oriented Linux solutions. In the same way, IBM NETVISTA THIN CLIENTS are provided with TurboLinux.
Our storage solutions ENTERPRISE STORAGE SERVER should also be considered.

paragraph **Software** IBM DB2 UNIVERSALE DATABASE has connectors for Linux. This allows the interrogation of DB2 bases on *mainframes* based on applications which work under Linux.

DOMINO LOTUS also works under Linux. It offers e-mail and *workflow* applications. A whole series of WebSphere type applicatives provide the Linux world with new e-business Java-oriented solutions. In particular, Mqseries which have been developed to accelerate e-business [7].

This offer does not neglect small and medium-sized businesses SMALL BUSINESS SUITE FOR LINUX [8] . Neither are developers forgotten with DEVELOPER KIT FOR LINUX which permits the development of java applications for xSeries and zSeries.

Services. The last part of the IBM Linux offer is services and support. *IBM Global Services* has developed an offer where services and expertise are available 24ha day and 7days a week. They concern Linux and are integrated into our traditional support interface. They offer a broad spectrum of training, adapted to our customers' knowledge and competence (from beginners to the most experienced). Our consultants are available

[5] To keep track of this offer, http://ibm.com/linux/ is proposed.
[6] On the iSeries, Linux completes OS/400 and does not replace it. OS/400 is essential to have a Linux partition work.
[7] http://ibm.com/software/ts/mqseries.
[8] http://ibm.com/linux/sbs.

to develop guidelines for those who wish to switch to the Free mode and find solutions which enable our customers to go from a decision-making process to production.

RNTL: *What is, precisely, the implication of IBM Global Services ?*

MARC JOLY: IBM Global Services has over 300 consultants trained to use Linux all around the world. They are deeply involved in what they do. They cover every single step for customers who want to switch to the Free mode. They advise and train them, they deal with strategy and tactics as well as deployment and support. Besides, our *Linux Technology Center* maintains the "Linux At IBM" Web pages, on http://ibm.com/linux, with comparatives, case studies, news, downloading and electronic support spaces. This Website can be considered as a contribution by IBM on behalf of the Linux community.

2.4 The community .

RNTL: *What kind of links can there be between the Free Software community and a huge organization such as IBM ?*

MARC JOLY: For the longest time, the community has asked IBM to get involved in Free Software in the same way it has asked other large companies that work in the same domain. We want Linux and our traditional offers to be complementary and we want this to be a success. We have no other choice but to integrate this community and accept its rules. This reinforces our credibility. Our implication in the Eclipse project [9]. We have also abandoned our "rights" on " FileSystem Management " [10].

Over 200 people in our *Linux Technology Center* work in.liaison.with the Free community.

It should be stressed that, undoubtedly, more than 80 % of those who work for the "Free community" actually work in businesses or universities. We also deal with them as customers. We thus do not necessarily see this "community" the same way an SSII would, since those who work in SSIIs often come from this community.

2.5 Economic model.

RNTL: *IBM does not seem to have a specific economic model with regard to Free Software : isn't this quite simply an application of the Free Software economic model, in miniature, within IBM Global Services ?*

MARC JOLY: Exactly. There is no specific economic model. There is IBM's global economic model and the Linux strategy described above which comes into play.

It is clear that the passion that Linux sparked in the business community is durable. IDC says that, in

[9] http://www.eclipse.org/, announced in 2001, whose objective is to develop and promote an open, joint development platform *open-source* and have as many developers and editors join it, is proof of this. The consortium gathers Borland, RedHat, Fujitsu, Serena, Sybase, Merant, SuSE, QNX, TogetherSoft, Rational, WebGain and of course IBM

[10] IBM is also involved in the "Journaled File System" initiatives.

2000, 27 % of the new installations of servers were done under Linux. This places Linux in second position. We estimate 40 % in 2005. We can see that software publishers are following this movement and have clearly understood their clients' interests : some 2000 (editors) now carry their solutions under Linux. Linux is not a market which is reserved to researchers, students or enthusiastic pioneers anymore. All sectors, from small and medium-sized businesses to very large companies are interested in it. And we clearly see this diversity through our customers' projects, ranging from the Comtesse du Barry to Opel, from the General Company of Geophysics to Aléos.

As I said earlier, Linux and IBM form a natural movement. We planned it more than two years ago and, today, it seems totally integrated into our offer.

RNTL: *Are the markets easily convinced ? Is the future of Free Software good in France ?*

MARC JOLY: Yes, it is very promising. Companies, but also administrations, are more and more interested in it. Some customers still remain to be convinced, but very few resist when we honestly promise a reduction of more than 50 % for their possession costs and promise to improve their service.

The future of Free Software is promising, but there will undoubtedly be concentrations. For example, there are over 200 distributions in the world. Over the long term, about fifty should be enough to meet our needs.

RNTL: *What kind of sales turnover does IBM make with its Linux solutions?*

MARC JOLY: Well, since we invested more than one billion dollars in 2001, you can imagine what we expect.

RNTL: *One year after this investment, what do you make of the situation?* MARC JOLY: This investment has enabled us to integrate Linux into our offer within a short time. Simultaneously, we were able to offer the services that go with it to help our customers to switch to Free Software.

- Part of this investment was devoted to the offer itself.
- Another part, to the integration of Linux into our products : iSeries, pSeries, xSeries and zSeries, DB2, Domino and WebSphere servers.
- The last part went to the offer for support and training, as in *Linux Technology Center* which has dedicated the services of nearly 200 people to Linux.

We can say that our objectives have been achieved.

RNTL: *Does the Linux strategy entail purchases or acquisition of holdings ?*

MARC JOLY: Not to my knowledge.

2.6 The necessary promotion.

RNTL: *How does IBM ensure the promotion of Free Software?*

MARC JOLY: In fact, there is not that much effort to make. Companies are eager to save money, and free software fulfills this need. Linux and our hardware infrastructures complement each other in a natural way. IBM's prestige draws the attention of the community of developers. And we are rather proud of the messages we convey through our ads, particularly our TV ads.

The "Linux cell" is a division of its own within IBM. It corresponds to a hundred people in Europe. Their activities range from servers to hardware, software and services.

RNTL: *You are in charge for Western Europe.*

MARC JOLY: France, Belgium and Luxembourg. I have my counterpart in Germany and in the Scandinavian countries. With such an important strategy, it is necessary to have people who implement it horizontally for IBM: servers, software and services. It is my role. Of course, this leads me to promote our strategy outside of IBM, through conferences, shows and on-line discussions, etc.

2.7 A story about men and women.

RNTL: *When the Linux strategy was implemented, did you hire people?*

MARC JOLY: Yes, mainly for services. The recruiting was done within the community of Free developers but also outside this community. We have technical gurus, as we call them. We use a policy of sponsorship : people paid by IBM who work for community projects.

RNTL: *How did this work with IBM business culture ?*

MARC JOLY: We mainly have people who come from very large systems. Their culture was not necessarily the same as ours. Our phase of observation of Linux and free software [11] coincided, in-house, with a phase of "awareness", and very quickly, our engineers were attracted by the Free development modes and the products which were already available. They are all convinced now of the cogency Free Software.

Finally, this proved to be an extraordinary advantage. We can release versions before everything else has been completed. Since at IBM, we do everything with passionate interest, Linux provides us with what we wanted.

2.8 IBM and the others.

RNTL: *What kind of relationship do you have with other Free actors ? Isn't IBM Global Services a tough*

[11] within IBM, a market development monitoring team worked upstream before IBM built its Linux offer.

competitor for SSLLs as they are sometimes called ?

MARC JOLY: As I said earlier, we work with several Linux distributions and not directly on the Linux kernel. Thus, it is not our objective to develop our own distribution. We work with MandrakeSoft around the xSeries.

Bull is involved in Linux and clearly is one of our competitors. This is something positive because the more people there will be on the Linux market, the more it will develop.

IBM Global Services is not a competitor. In fact, we are driven to work with the French Linux software companies [12].

IBM Global Services hosts on Linux-oriented *mainframe* , but WorldCom also proposes its offer.

IBM has also assisted the young Aleos startup in the supply of hosted applications (a S/390 (zSeries system), a storage and safeguard system, where another solution would have required 100 or 200 machines). And it is not IBM, but Overlap, one of the *mainframes* partners in France, which dealt with the implementation of the platform which had been selected. And, in the same frame of mind, Aleos reaches out to companies specialized in Free software such as Alcôve, Aurora, IdealX or Linagora.

IBM also helps software publishers, with a portage center in the Paris area.

We cannot go against Linux SSIIs. They are close to the community and we wish to be close to them too. They have credibility and, together, we reinforce the credibility of Linux solutions. They have their expertise and we have ours. Together, we form a natural and winning team.

RNTL: *It is appropriate to add that Marc Joly was questioned on IBM licenses (http://www.opensource.org/licenses/ibmpl.html) and did not elaborate on the matter. On the other hand, he stressed the 3 year warranties that come with the z900, for example.*

[12] Just as well, testers were able to see the solutions we had planned on using, before the deployment of our Linux strategy (Cf. introductory section), just as well, these tests are now carried out in-house and by our partners at the same time.

AYMERIC POULAIN-MAUBANT,
Jipo Interactive, Aymeric.PoulainMaubant@jipo.com.

The editors: ACT Europe.

Summary.

ACT Europe is a major actor in the Ada 95 sector. It offers both products and services. Recognized as such, it has been present on more platforms than its competitors and provides unequalled user support. It thus works with clients who develop critical applications in many different domains : aerospace science, defense, energy, transportation, banks, media and communication. Its flagship product is GNAT Pro, development environment for Ada 95. This environment, based on the —free— GNU/gcc technology, is available for both native or embedded applications. It comes with a whole set of peripheral tools, libraries and documentation. However, this tool cannot describe ACT Europe activities. Indeed, a service offer comes with the product. It allows users to work with ACT Europe engineers who will be partners and will play a central role in the implementation of the client's projects. The client team and ACT Europe work together. This allows them to better define the Ada tools which are necessary and to provide the client, throughout the entire project, with the help he/she needs in the field of software development. This co-operation has immediate effects which are beneficial for both the client concerned and for other clients: optimization and code organization, language evolution, multi-language systems, but also less risks and higher productivity, *time to market* reduced. Corollary: all products come with the sources. This does not mean that proprietary or closed software cannot be developed one way or another. Second corollary: ACT Europeactively participates in the effort of the Free community : it greatly contributes in both, the GNU project and GNOME environment.

- Person interviewed: Franco Gasperoni
- and reference site: http://www.act-europe.com/http://libre.act-europe.fr/

3.1 A few reminders.

RNTL: *What is ACT Europe?*

FRANCO GASPERONI: ACT Europe is a unique and slightly different company. It was founded in 1996 by both *Ada Core Technologies Inc.* and the European members of the GNAT Ada 95 project. It provides industrial, classic or military clients, in Europe, with top support that evolves around the GNAT Pro Ada 95 development environment.
Its objective consists, in fact, in providing the European Ada community with the highest level of expertise concerning Ada software and, consequently, the best technical support. [1].
Ada Core Technologies was founded in 1994 by the main authors of the GNAT Ada 95 compiler. However, its expertise dates as far back as 1979, when Ada was created. In 1983, the first validated Ada system showed concrete proof of this expertise. Its preliminary work was conducted at *New York University* by a team which was entirely assigned to Ada Core Technologies in order to keep on insuring the success of the Ada language and make sure that its users were still offered high-tech technologies.

RNTL: *As one can see, Ada 95 is at the heart of your activity. What are its characteristics?*

FRANCO GASPERONI: Ada 95 is an object-oriented programming language, ISO/ANSI standardized and compatible with its predecessor, Ada 83. At the beginning, the latter had been developed for highly critical and ambitious applications, in particular, for embedded applications. Thanks to its qualities, it was adopted by material builders whose list gives an idea of what they expected from such a language: Boeing (777) and

[1] the term community should not be interpreted as in «Free software community ». But we use it and keep it here because ACT Europeand Ada more generally, through their university background, have a special relationship with Ada users, even if the latter could be considered as typical closed and proprietary software users by external observers.

Airbus (A340), TGV, the entirely automated Madeleine-BNF Paris metro line and even systems that are taken on board satellites.
Ada 95 is an extension of the first Ada 83 draft [2] which offers a single combination of characteristics which are object and real time-oriented. It is also oriented towards competitive and distributed programming approaches. Such a language is ideal for complex software projects (several million code lines), which is accomplished over the long term, which must be highly maintainable, have reusable and extensible codes and which concern critical missions for which computer systems must simply be totally reliable.

RNTL: *You also mentioned the GNAT term several times.*

FRANCO GASPERONI: GNAT is Ada software (95) most widespread development system. It combines both Ada strong points and GCC "open philosophy" [3]. In addition to qualities one can expect from a compiler, it was developed and promoted to attract new Ada users and to make sure that, whenever a project required efficient and safe codes, Ada would be chosen. Our GNAT Pro product is available on a broad spectrum of platforms, PC workstations and bare board cards [4] for conventional or advanced applications - real time, embedded, distributed —, broad (millions of code lines) or modest (tens of thousands).

GNAT is based on the GNU – free – environment. Thus, the compiler itself is integrated in GCC. The debugger is based on GDB. GLIDE development environment is based on Emacs. GTKAda. GUI is based on GTK environment and offers advanced construction functionalities for graphic interfaces. In fact, each one of these products (and these are only a few examples) is based on a former free GNU "offer" and proposes advanced functionalities which are adapted to Ada and the needs of its users.

3.2 Strategy.

RNTL: *You have just explained both your background and your job. Now, how would you describe your strategy?*

FRANCO GASPERONI: To begin with, beyond the general objective which is to provide the most advanced Ada products and services, we have established a certain number of partnerships with materials as well as software sellers. As far as hardware is concerned, we can mention the relationship we have with ACT and Silicon Graphics Inc. whose SGI Ada 95 products are based on GNAT. We also work with Hewlett Packard and Compaq. As for software, we work with Aerospace science, British Aerospace, Construccionnes Aeronauticas in Spain, Lockheed, BNP, Canal+, Ericsson, Nortel, Philips Semiconductor, the US Air Force Academy and West Point. This gives you an idea of the sectors we deal with.

RNTL: *You talk about a "partnership". The idea is to establish a long-term relationship with your customers...*

[2] The latter was characterized by a strong idea: to clearly separate the interface (the definition of functionalities), and the effective implementation.

[3] GCC is a compiler, that is a software program which transforms a source code into a readable code by the target processor. It is also a free software program per se. It produces high quality readable code for many target processors, which ensures great portability to the languages that it supports (C, C++, Ada, Java in particular). It also allows crossed compilation, that is to say, it compiles codes for platforms which are different from the ones it is working on. This, in turn, allows great flexibility for developers.

[4] *bare boards* .

FRANCO GASPERONI: Exactly. There is no question of selling a product to a customer and letting him/her handle the rest on his/her own. Support is at the heart of our activity.

In the proprietary mode, selling the product is the number one priority. This entails huge marketing costs. In this frame of mind, the support that is offered is a burden for the salesman, whereas, at the same time, users and developers need more and more assistance to better use the products they buy.

We believe that, from the moment one seeks the best equation between quality and *the time-to-market* , support is the key factor. Nowadays, technology creates such a level of complexity and information that support is what makes it possible to approach this complexity in a serene way. This is why it is at the heart of our activity. Support for customer-projects which have a lifespan of several decades...

RNTL: *How do you implement this support on a daily basis?*

FRANCO GASPERONI: We can come back to this later in more detail, but in a few words, we receive questions on how to use Ada, explanations on how to use our tools effectively, report analyses on such or such a problem, questions on specific developments and specific improvements, we offer either regular appointments or a 7-day assistance.

RNTL: *What does the free software mode bring?*

FRANCO GASPERONI: It makes all the difference.

We believe that the Ada world benefits from a non-proprietary technology. And we do deliver all our products with their sources.

Our customers must provide reliable software applications, which must remain reliable for many years. They cannot rely on "black box" type of software offers and editors who can modify the characteristics of their products in a undesirable manner. Moreover, they are not expected to be experts in all domains of a language and, in particular, its core. They must indeed concentrate on the terms and conditions of the software they develop and on business of their own customers (and their own, perhaps).

The open way in which we function guarantees them permanent control of all the components of their final application. They can also reach the developers who have built these components at anytime.

Another immediate benefit, which also constitutes a significant share of our strategy, is to allow the emergence of *pool* pools of expertise. We believe that the best experts work for us, but competition exists: qualified people who also know GNAT well, some of them work for or with our customers.

Lastly, and this is a guarantee for us and for our customers, our open codes are read over and over again, carefully examined by a wide community of Ada experts, in particular in universities. We also work hard so that the judgment of our peers will always be positive and that they appreciate what we produce: the software capacities as well as the way it has been written.

3.3 Products and services offered.

RNTL: *Let us come back to your offer. How does it work?*

FRANCO GASPERONI: As far as products are concerned, we offer:

- *GNAT Pro*, a complete Ada 95 development environment. This environment includes the compilation chain, the advanced glide editor and the debugger
- *GNAT Pro High-Integrity Edition* specialized for the highly critical systems item *GNAT Pro Bare Board*
- *GNAT Pro for Open VMS*
- *GNAT Pro for VxWorks*

to which we will add :

- CGNAT, GNU C
- GtkAda development environment which allows the development of modern graphic interfaces for X-Window and Windows
- JGNAT, Ada for Java
- GLADE platforms, GNAT in
- GNATCOM distributed systems which allow the development of CON+/DCOM/ActiveX components in Ada
- ASIS-for-GNAT applications in order to develop on-tools[5] Ada
- XML/Ada to handle XML flows XML in Ada

applications.

RNTL: *All these products come with their support.*

FRANCO GASPERONI: Yes. All these products come with their support. They come in a variety of forms:

- GNAT Partner, link between the customer team and ACT Europe engineers;
- GNAT Enterprise, for either refined customized projects or projects which must be achieved within tight deadlines;
- GNAT 24/7, whose name gives an idea of the extent of support that is offered;
- GNAT Assurance: the code produced by the customer is integrated in the quality approach of ACT Europe.

Support also consists of traditional advice and on-site training.

RNTL: *You said that ACT Europe was unique in its own way.*

FRANCO GASPERONI: Yes, because, as you have seen, our clients' projects receive support which is central to us. Our engineers take part in it. In this way, development teams are combined.

[5] Allowing, for example, the creation of Ada software test or measurement tools.

This personal approach is the insurance that the customer will use our products in an optimal way and that he/she will be assisted in all the phases and for all aspects of his/her development. In addition to this exchange concerning the project of the customer, our teams also deal with the optimization and organization of the code, the evolution of the language, the aspects related to the multi-language systems. Following this network logic, the improvements that have been realized for one customer can be beneficial for everyone. These principles lower risks, ensure higher productivity, and a reduced *time to market*.

Typically, all our products come with project support which is an annual contract containing:

- the software CD-Rom for the target platform, as well as documentation (written on paper),
- a one-year FTP access so as to update the product and its documentation,
- a one-year subscription to two lists of user groups, ug@gnat.com and ug@act-europe.fr,
- one year of " GNAT Product Support " and one year of "GNAT Project Support".

RNTL: *And what can be found in these two support offers – Product and Project —?*

FRANCO GASPERONI: For Product, assistance for the installation procedures, explanations concerning the functionalities which are related to the implementation which has been chosen, the correction of the problems which may have been encountered during the utilization of the product, an extranet to present problem reports and have access to pre-versions of various products.

For Project, a more sustained type of assistance of all aspects concerning the development under Ada, including the differences between Ada 83 and Ada 95, advice on how to best use Ada 95, GNAT Pro and the tools which accompany it, how to configure them, how to migrate from a non Ada technology towards Ada, how to link Ada and C or C++.

In the same way, all "add on" products, such as CGNAT, GtkAda... have this type of offer, adapted to their characteristics.

RNTL: *These are, what one might call, traditional support offers...*

FRANCO GASPERONI: ... to which we added the Partner, Enterprise, 24/7 and Insurance support as I said earlier.

RNTL: *Then, what are the characteristics that make them different with more general kinds of support?*

FRANCO GASPERONI: First of all, all these contracts are yearly contracts and are attached to only one project at a time.

With GNAT Partner, a senior ACT technician represents the project of the customer within our team. This is someone the customer can talk to directly, who analyzes the various problems that he/she meets with him/her, and who keeps him/her posted on GNAT technology latest developments. This support comes with a 2-day on-site training and with regular telephone appointments.

GNAT Enterprise includes the GNAT Partner offer, with an extra 3-day training and advice. It is an offer which is more adapted to tight deadline projects *deadlines*, or which require some personalization and wider GNAT functionalities. 3 persons/week are thus included in this support : they either make the desired improvements or go and see the customer in order to solve points that the team could not have solved from a distance.

GNAT 24/7 extends the GNAT Enterprise offer: it includes a continuous telephone assistance and a 90 minute guaranteed answering time.

Finally, GNAT Insurance is a special support offer which places our customers' applications within our own internal process of quality insurance. Thus, the products of the customers who have subscribed to such support are included in our regression tests, so that the future versions of GNAT Pro will affect neither the behavior nor the performance of these products. In addition, as soon as a customer encounters a problem, he/she does not have to isolate the problem on his/her own: he/she tells us what he/she observes, and it is for us to precisely isolate the problem and the code section that is concerned.

RNTL: *In addition to this broad support offer, you also propose services.*

FRANCO GASPERONI: Such as portage on specific environments, personalization of our tools or the addition of functionalities. But we also offer on-site and off-site advice and training (between one and five days).

3.4 Economic model.

RNTL: *Let us speak now about your economic model. To start, you have a university background.*

FRANCO GASPERONI: Yes. Back in 1994, as far as the States is concerned, and in 1996 for Europe. This was due to an increasing demand for a commercial support around GNAT. We started with some classic, small loans and have survived since the beginning thanks only to the income from our activities. This income was a lot higher than we would have imagined in the beginning, and this made ACT quite a profitable company.

More precisely, since most of our income comes from our support offers, it is our duty to ensure the best assistance to our customers. On the other hand, we must have a renewal rate of at least 90 %, the remaining 10 % correspond to projects which are entirely finished.

This approach enables us, year after year, to have a stable and predictable flow of income. This can last indefinitely [6].

In addition, more than 90 % of the personnel are top technicians, the remaining 10 % consist of a small marketing and commercial team. This allows the company to have low overheads. We can thus satisfy ourselves with a reasonable flow of income while maintaining high quality offers.

RNTL: *What are your fees?*

[6] As long as Ada technology keeps on growing, of course. However, it should be recalled that ACT does its best to improve GNAT and to make it evolve according to the customers' requirements and thus it always keeps this system on the leading edge. And this is done precisely through support offers.

FRANCO GASPERONI: I will say that "prices have nothing to do with free software". What matters most is the freedom that is offered with the source code. Whether you pay or not to have a copy of our free software, what you have is the freedom to copy it, modify it and redistribute it.

Thus, in a natural way, the fee can simply not exist. In the proprietary mode, you pay a license to use a software, then you pay for the maintenance. This is the case for GNAT, with a zero operating license. This does not mean that you do not have to pay to use GNAT. Especially for huge software projects which necessarily imply support, if possible, from those who are the original developers of the software.

Let me take this opportunity to make a brief statement. This business about free license has immediate advantages for everybody, for us and the customers who are going to ask for our support. Since our software programs are available to everybody, one finds on the market a growing community of users and developers who master Ada and GNAT better and better. This enthusiastic community offers major contributions in order to improve its products. Moreover, since they are free of access, our products can be intensively tested, without any restriction, before being definitively adopted.

And none of this destroys the need for an offer of support.

RNTL: *What is your sales turnover?*

FRANCO GASPERONI: Between 3 and 5 million Euros. That is five times less than if the system were under a proprietary mode.

RNTL: *Have some customers specifically asked you to develop under a Free software mode?*

FRANCO GASPERONI: Yes. There are some customers who consider this to be their number one criterion. It gives them the guarantee that know-how will be available through many different people and not only through a small group of a dozen of experts.

RNTL: *Can such an economic model work when competitors can easily take the source codes of your software?*

FRANCO GASPERONI: For the time being, this is not a problem.

Besides, competitors can do that in an illegal way (as understood by the GPL, that is to say, they put their whole production under a proprietary mode. This includes what they obtained under the Free mode). In a way, this is not too upsetting, because it is always better to have some form of competition, not to be the only one on the market.

Anyway, people should stop thinking systematically in terms of market shares. We are not limited to this logic anymore. In our environment, we think more in terms of "give and take" rather than in accordance with traditional economic logic. Therefore, we are much more interested in the trust the entire community of users shows us than in the "market shares" our competitors can obtain through our codes. It is another ecosystem, another balance, in which initial creators [7] have an advantage at the beginning. They can preserve it as long as they strongly commit themselves to both innovation and close customer support.

ACT is, and remains the technical leader for everything that has to do with Ada 95 development. Admittedly, I did talk a lot about our support offer, but that does not prevent us from investing in a considerable way in continuous development. This places us well ahead of our competitors. We are thus the only ones to

[7] Note that the 5 initial founders are still present at ACT Europe.

offer support on the entire Ada 95 system, including its appendices (*information systems, safety and security, distribution*), as well as the versions for VMS, Linux, OS/2 and Solaris x86.

The market we are aiming at clearly helps us. Such a thing would be more difficult on a market intended for the general public. Here, we can remain honest and committed engineers, without having to worry about complicated marketing strategies.

Therefore, as far as we are concerned, this system works.

3.5 A story about men and women.

RNTL: *To conclude, some words about the human aspects of this experience. What kind of relationship do you have with the community?*

FRANCO GASPERONI: We are strongly committed to the community [8]. This shows up in a variety of forms on our community Web site http://libre.act-europe.fr/. We offer tools for developers as well as for professors and students. It is for this second category, in particular, that an easily accessible community site, is important. It is from this site that people can download complete versions of our products, but without the support of course.

RNTL: *And within ACT Europe, how are the people managed?*

FRANCO GASPERONI: Quite simply , it is an organization which reflects the principles of the Free software community. We have a non-hierarchical structure. We work with mature and responsible people who are very independent. Some of our employees have had interesting ideas and have developed them on their own. ACT Europe then showed interest in them, and, now, they are part of the range of things ACT Europe offers. It is the case, for example, with all the offers around XML.

3.6 As a conclusion.

RNTL: *Do you have a few final words to conclude?*

FRANCO GASPERONI: ACT Europe is a company which has three vocations. On one hand, it is devoted to its customers, who have complex projects, over 5, 10 sometimes 20 years. It is devoted to Ada, so as to make Ada accessible. And it is devoted to free software, whether it can be used on free systems or not.

[8] NDLR: It should be noted that, once in a while, what one could call small community cores, appear in a spontaneous way, in particular within the scientific community. One might think that part of the community is embodied there, rather than seeing it in a diffuse way at the heart of Internet. Ada, with its university origins, probably explains these phenomena.

The editors: MandrakeSoft.

4.1 MandrakeSoft.

- Person interviewed: Jacques Le Marois, CEO and co-founder
- reference site: http://www.mandrakesoft.com/

Avertissement: The Linux-Mandrake[1] site was selected among the top 100 Linux sites chosen in June 2000 by Linux Magazine. It was awarded the *Editor's choice* e-managing from the *Open Directory Project*. It clearly deserves this recognition, because this site (like many others, as we will see below) is both clear and complete with regard to the type of information people expect from such a commercial site. But it even goes further than that: it gives ample information concerning how the company works (its introduction on the stock market) and its commitments. In a purely free software spirit (as far as borrowing, adapting, quoting one's sources are concerned), we decided not to reinvent the wheel and, consequently, we chose to use large portions of this site as long as they were relevant.

4.1.1 Activity.

Founded in 1998, MandrakeSoft is a computer facility/service company which is specialized in the use of the Linux operating system. Its activities are organized around 2 poles:

- *tools* (97 % of the turnover). Utilities make it easier to install, configure and run the Linux operating system nucleus. This pole can be divided as follows:
  - *retail sales* through a network of wholesalers and distributors (68 % of the activity right now versus 78 % in 2001)
  - *ventes online* on the site. MandrakeStore[2], more lucrative in terms of gross margin (28 % of the activity right now versus 9 % over a 4 month activity in 2001)
  - *OEM sales* concerning 3 % of the turnover
- *services* (3 %): Website and telephone support with limited access.

As for the different geographical zones, the turnover is spread out as follows: USA (73 %), Europe (25 %) other (2 %).
MandrakeSoft edits the Mandrake Linux operating system following the open development approach (free access to the source code which is published according to the terms of the General Public License). Around the world, several hundred developers continue to contribute to improving the product via Internet. Mandrake Linux is the most widely used Linux distribution system on the planet: it comes in more than 50 different languages (when installed). Mandrake Linux includes high-performance graphic interfaces which have been pre-configured (particularly KDE and Gnome) and over 2300 applications. These include the famous Netscape Communicator, StarOffice (automation) and the Apache Web server. MandrakeSoft collaborators are found in more than 20 countries. However, the company itself is physically present in France, the USA, Canada and Great Britain only. Since July 30, the company has been registered on the Euronext Free Market, Paris (code Euroclear: 4477.PA MandrakeSoft/Boursorama [2002]).
MandrakeSoft defines itself as both a project organizer and competence unifier in the Open Source field. Its ambition is to help its users benefit from innovations that derive from the work of the community, which unites several hundreds of individuals around the world. It proposes a powerful and stable operating system which is associated with a complete range of solutions for client stations (graphic environment such as Windows,

[1] http://www.linux-mandrake.com/.
[2] http://www.mandrakestore.com/.

automation which is compatible with any type of file format, Internet browser) and for servers (Website server, mail server, file and printing servers...).
Today, Mandrake Linux is the distribution[3] Linux comes in 50 different installation languages and is the most widely known around the world. Moreover, Linux-Mandrake guarantees simple, fast and reliable installation and support. Linux-Mandrake offers different solutions: they include the operating system, the programs which are associated with CD-ROMs, installation tools, complete documentation as well as support. As far as services are concerned, MandrakeSoft goes even further and offers backup assistance to its clients (as well as individuals users, associations, organizations and companies): these are either package solutions and / or customized [4]olutions particularly for State organizations such as the Ministry of Culture or the Air Force..
MandrakeSoft claims that several million individuals and several thousand companies use it.

4.1.2 History.

MandrakeSoft was created in November 1998 thanks to 3 young men who had a passion for Linux operating system and who met on Internet: Gaël Duval, Frédéric Bastok et Jacques Le Marois. A few months before, Gaël Duval had placed a first version of the Mandrake distribution on the Internet. It was based on the equally famous RedHat which was a big hit right away. Special efforts had been made on the installation system. Easily downloaded, the brand-new company quickly spread all over the world. As early as February 99, a new version *packaged*, with a user's manual, was made available. This success attracted partners, particularly a rather important American network distribution group (Macmillan Software, recently named Pearson Technology Group). This allowed the company to establish itself in the United States on a long-term basis (in September 1999, a subsidiary opened there).
All through 1999, the company was given numerous awards. This consolidated the respect users had for the company and which was not contradicted in 2000. Within two years, the French company had imposed itself and had become the second global reference as regards Open Source and Linux software programs. 2000 was a year of economic growth: offices and subsidiaries open in London and in Canada). New partnership agreements were signed (value added dealers and training centers).
In 2001, portfolio offers continued increasing which had been created the year before with package offers such as training and technical support. Now, the company was opening on-line associated community platforms as well as on-line shopping platforms. Moreover, OEM agreements [5] were signed with Compaq and HP. Finally, the most significant event that took place in 2001 was the company's stock market listing (Paris Euronext Free Market). We will come back to this point later on.

4.1.3 Products.

MandrakeSoft mission is simple: to propose a global and innovative offer which includes Open Source products as well as services which are adapted to its users. This way, they can benefit from the output, performance and simplicity of Mandrake Linux operating system.

- *Retail Market* At the beginning of 2002, the offer consisted in:
  - Mandrake Linux Standard edition 8.1. It is conceived for both beginners and experienced users. It offers the basic tools to discover and use Linux solutions.

[3]There is a difference between «distribution» and mere «assembly». The Mandrake product is more than a simple combination of free software programs. It offers an installation and confi guration system which has been especially developed and which has made the product successful. Many users compare this distribution to a Swiss knife because it is so easily adjustable.
[4]s
[5]*Original Equipment Manufacturer*.

  - Mandrake Linux PowerPack edition 8.1, *pack*. It comes with a complete range of over 2300 applications on 7 CDs. It also includes a 30 day hotline assistance and 60 days of on-line help throught the Internet.

- *Business Market (SMEs, Large Companies)* At the beginning of 2002, the offer consisted in Mandrake Linux ProSuite edition 8.1. It is conceived for small and medium-sized businesses but is easily adaptable so as to be used by large companies. It offers broad technical support. It puts the stress on the tools that are necessary for the implementation of a complete computer environment. It also guarantees stable and safe solutions for:

- *OEM Product* servers and work stations. Based on Mandrake Linux Desktop, the OEM product was especially elaborated so that computer manufacturers could propose material and software turnkey solutions to the final user: the pre-installation of Mandrake Linux on manufacturers' material along with technical support.

- *ISV Offer*[6]. The ISV offer applies to software editors and developers who, thanks to the Mandrake flagship, benefit from the Linux market's fast growth. In fact, it is a partnership program which gives ISV products greater visibility thanks to, for example, the software solutions which can be found in the Mandrake catalog (Solutions Kit). This partnership also includes a complete technical and commercial collaboration and a high-performance technical support tools.

It is important to stress that, since the creation of the first version of Mandrake Linux, MandrakeSoft has kept working on technological innovations. It does not simply assemble software programs to distribute them, but it endeavors to create the technological binders (for example, the installation systems) which make it a user-friendly tool and which have contributed to its reputation. Its compatibility with the « parent-distribution », RedHat, had to be maintained. For example, the following tools have benefited from the expertise of both, MandrakeSoft Research and development teams and Linux-Mandrake international contributors. The Linux non-specialist user will appreciate the efforts made by the company's R&D department as well as the community's involvement.

- *DrakX* is a tool whose role is to facilitate the installation of Mandrake Linux distribution from a CD-Rom, the network or a hard disk while utilizing a graphic and user-friendly interface.

- *Lnx4Win* handles the installation of Linux on a Windows partition (95 or 98).

- *DiskDrake*, hard disk partitioning utility for PCs whose role consists in dividing storage units into several parts thanks to the mouse. Partition information is then presented in a visual interface. This functionality is integrated into DrakX.

- *HardDrake*, an ambitious project which aims at facilitating, through the operating system, PC internal peripheral recognition as well as their configuration. Here again, a graphic interface makes things easier for the user.

- *RpmDrake* makes it easier to either install or suppress software programs.

- *DrakConf* handles peripheral «post-configuration».

4.1.4 The consulting offer.

MandrakeSoft does not content itself with offering products. It adds business-targeted counseling for its products. Thus, it can help companies build their projects thanks to an appropriate solution based on its experience

[6] Independent Software Vendors

as an editor: safe computer environment, migration towards Linux, customized distributions, Open Source solutions (Website servers, mail, file sharing...). This counseling can be provided by using three different approaches: a project, a package deal or line support. They can be proposed along with training and technical support if need be. Moreover, MandrakeSoft partners are usually involved in this counseling and can offer an even more personalized approach.
This counseling offer is even treated at an earlier stage [7] and in a non-stop manner [8] through the classic example, Website: http://www.mandrakebizcases.com/

4.1.5 The training offer.

Following this same logic, MandrakeSoft offers off-line and on-line training. This training is dispensed in centers and responds to the needs of both, the final users and system administrators. Computing courses also offer constant up-dating on Linux solutions. Linux-Campus training program is available through a network of certified partners Centers. Linux-Campus is a reference in the world of Linux training. Its fame is even reinforced thanks to its partnership with Linux Professional Institute (LPI) which offers a professional Linux certification which is recognized worldwide. http://www.mandrakecampus.com/ MandrakeCampus goal is to provide the GNU-Linux community, from new users to specialists, training courses and classes on everything that deals with free software. Thanks to this electronic platform, teachers, tutors and learners profit from a forum where they can share their knowledge. So as to respect the Open Source mindset, which is what makes MandrakeSoft and its community of users so strong, Mandrake and Logidée are finalizing an agreement which will enable Mandrake-Campus.com to put all the information it has under GFDL license (GNU Free Documentation License). It is worth noting the effort that has been made in creating on-line tutorials which entirely based on high quality images.

4.1.6 The support offer.

Over the long term, counseling and training might not suffice. That is why MandrakeSoft also proposes to provide different types of support in addition to adequate forums where users can exchange their knowledge. MandrakeSoft proposes a whole set of professional services to businesses. These include high-performance support which ensures fast installation and configuration of a complete professional environment. Otherwise, Mandrake Linux proposes another Website which is available to all users: http://www.mandrakeexpert.com/ this is an on-line platform which allows users to ask questions and to obtain answers from experts. One can, alternately, play the role of an expert or that of a regular user who needs to obtain some information.
Of course, each Mandrake Linux pack comes with free support which can be activated once the pack[9]as been recorded this allows Mandrake to partially follow the dissemination of its products. This is known to be something particularly hard to do with free software and which has remained a weak point for a long time if we compare this with pre-installed proprietary software editors who claim a number of users which is obviously over-estimated.

4.1.7 Competitors.

Two kinds of competitors (who, furthermore, share points in common: distributions (Debian, Redhat...) and distribution resellers (and/or associated services). This second type of competition is rather common among businesses, whereas the first one appeals to people's emotions. These emotions derive from the RedHat com-

[7] that is to say that this Website enables prospective customers to find the first answers to their questions and to quickly judge the quality of Mandrake advice.
[8] as new testimony comes in, the site is improved.
[9] h

munity and distribution. They still strongly determine the clients' choice and influence the community's opinion toward Mandrake and toward its own strategic choices (stock exchange, invitation to subscribe to the users'club, see infra.). It would be necessary to make the history of the distributions[10] so as to better understand these emotional data.
This having been said, several recent polls show that Mandrake Linux has succeeded in becoming one of the most widely used Linux versions in the world, in less than three years: These polls were taken among individu-

Linux Magazine UK	Linux.ie (Ireland) - July 2001[a]	PCMag.com - 8 October 2001 - US[b]	Linux.com (US), September 24, 2001[c]	LinuxUser Magazin - November 2001 (Germany)
Exhaustive study based on the opinion of 35.000 readers	Which distribution do you like best?	Which distribution do you like best?	Which is the best distribution for a beginner?	Which distribution do you use regularly?
Mandrake: 46.02 % Red Hat: 21.33 % SuSE: 18.67 % Debian: 5.33 % Corel: 2.66 % Caldera: 2.66 % Others: 3.33 %	Mandrake: 35 % Debian: 23 % Redhat: 18 % SuSE: 18 % Caldera: 1 % Corel: 1 % Others: 1 %	Mandrake: 43,78 % Redhat: 27,9 % SuSE: 15,41 % Libranet: 1,42 % Other: 11,49 %	Mandrake - 36,7 % Redhat: 33,5 % SuSE: 13,5 % Slackware: 5,6 % Debian: 4,6 % Caldera - 3,7 %	SuSE: 76,2 % Mandrake: 33,8 % Redhat: 19,7 % Debian: 10,7 % Slackware: 2,8 % Caldera: 2,4 % EasyLinux: 2,1 % Others: 2,8 %

[a] http://www.linux.ie/polls/pollresults.php?recount_pollId=1
[b] http://www.pcmag.com/poll_archive/0,3044,p~\%253D1035~\%2526bn~\%253D1,00.asp
[c] http://www.linux.com/polls/index.phtml?pid=128

als, not among companies. It is thanks to the work it has done to easily install and use Linux that MandrakeSoft has been so successful. The high number of users is a big plus in helping MandrakeSoft develop in businesses. Every satisfied user is a potential prescriber.

4.1.8 Explications for a success.

Several factors explain the exceptional growth of the company. The number of its employees has risen from 3 to more than 120 within 2 years. Following the «Open Source» philosophy, MandrakeSoft puts its Linux-Mandrake source code on the Web, so that anybody can have access to it. Hundreds of independent developers around the world keep improving the product. As a consequence, Linux solutions are among the most simple, complete and powerful to use. The company's economic model partially lies in the rapidity with which it has developed worldwide. The consequence of this is that Linux-Mandrake is the most internationalized distribution if one takes into account the origin of its contributors. This allows it to spread as rapidly as possible. Moreover, the company is settled in France, in the US, in Quebec and in England. Specialists do recognize the technological excellence of MandrakeSoft products. Linux-Mandrake was named «Best product of the year» and «Best server distribution» during the 1999 LinuxWorld Expo. In April 2000, it received the «Platinum Award» from the British magazine PC Answers as well as the Editor Choice Award from the famous American Linux Magazine in September 2000. Number 1 in France, Linux-Mandrake was also number 1 in the

[10] based on http://www.linux.org/dist/list.html.

States as far as selling Linux products in 2000. It did so for three months in a row with more than 30 % of the North American market (Source: PC Data). MandrakeSoft's success shows that the American software industry can be easily challenged directly from France. MandrakeSoft was registered as an «innovative company» by Anvar. It was one of the first distribution editing companies to be given such an approval for its Open Source software expertise. Finally, MandrakeSoft is strongly involved in numerous actions designed to defend and promote both free software in general and Open Source software in particular.
This success has attracted many investors [11], partners [12]. It has also allowed the creation of strategic and technological alliances[13], not to mention partnerships with different universities [14]

4.2 Possition in the Community.

MandrakeSoft has always seen to make Open Source a strong community. Thanks to its contributions, it favors the constant development of its company as well as that of its software offer. Indeed, it has everything to gain from a close-knit and better informed community. A product of the community, Mandrake regularly carries out actions which follow such logic. This logic is dear to the company's collaborators and aims to:

- *Contribute to making the community stronger*[15].
  - It is constantly involved in community organizations and movements (FSF, Linux Standard Base, KDE League, GNOME Foundation...)
  - . It actively supports open-source projects and open-source independent contributors (Gnome, KDE, Kernel, KOffice, Plex86).
  - . It implements internet tools for the use of the community (the lodging of sites, MandrakeExpert, PHP-Nuke).
- *Enlarge the Community.*
  - This is accomplished by broadening the community in order to permit the development of partnerships so that Linux can be massively diffused,
  - by supporting LUG actions (Linux users groups) as well as those who organize events in order to promote free software,
  - by allowing the greatest number of people to have access to knowledge thanks to free software (projects against a digital split or *Digital Divide*).
- *Internally*. Concrete gestures show how much the company is attached to such values
  - A choice has been made to develop its products in an open way with Cooker[16]

[11] AXA Placement Innovation, Viventures (Groupe Vivendi), Groupe Iliad, BBS Finance, France Innovation (ABN AMRO), Azeo Ventures (Groupe Lazard)...

[12] distributors: ABC Analog, ACI, Art Net, CHS/Métrologie, Editions Profi l, ID Pro, Info Networks, Isyotech, Italsel, Kangaroot, Lehmanns, Linux Discount, Macmillan USA, MCD2, Prisma Opentech, Softline; and service providers: Alcôve, Hewlett Packard, IBM Global Services, IdealX, Savoir Faire Linux, Sun as far as training is concerned

[13] Alpha Processor Inc., Arkeia, AR Systèmes, ASL, Aurora, Bull, Hewlett Packard, IBM, Matra Datavision (subsidiary of Aérospatiale Matra), MaxSpeed, Quadratec, Solsoft, Sun Microsystems...

[14] in particular with ENSTA within the framework of a teaching project for the development of Open Source software.

[15] MandrakeSoft supports numerous organizations and free software projects: FSF Europe, GNOME Foundation, KDE League, KDE, Plex86, Bastille Linux, Prélude, RpmLint, Urpmi, PHPNuke, Linux Kernel Development.

[16] Following the Open Development principle which has made Linux so successful, MandrakeSoft has put a pilot version of its future distributions called Cooker at the disposal of both independent programmers and independent users. This project, as well as the concept of putting together ideas and programs whose goal is to build an extremely a high-performance distribution, have attracted several hundred volunteer developers. Once again, this type of project shows how much the members of the Linux community are determined to work together.

  - Our developments are systematically put under GPL license
  - Buying up a company so as to put influential products licenses (plex 86) under a GPL license for the benefit of the community

- *LUGs*. Finally, MandrakeSoft makes contacts with local groups of Linux users on each continent

Each Mandrake Website (several URLs were mentioned above) contributes to making this notion of community even stronger, and one of them: http://www.mandrakeforum.com/ is specifically designed to spread the information which is linked to the Mandrake distribution or to the entire community.
In the same way, http://www.mandrakeuser.com/ stops the Mandrake user from remaining isolated, whether he/she wants to be in contact with the other users or not (on-line documentation and on-line discussions).
The community which evolves around Mandrake is composed of both, individuals and groups of users (the first purpose of the Linux community) and partner companies.

4.3 Strategy.

MandrakeSoft's strategy is based on the promotion of Linux among all of its users, from the beginner to the expert, from the private individual to the professional. In order to do so, it has elaborated a complete and customized range of services around its Linux products.
This strategy also consists in defending certain values: quality (in particular with regard to installation and configuration utilities, the instantaneous detection of additional peripherals, ergonomic interface) and the contribution it can make to the community. These values are stamped with legitimacy and sincerity[17]. Not too many companies can argue such facts. Indeed, they often behave in an opportunistic manner: they have tried to « make it » in the field of free software (parasites) but quickly had to give up.
In the months to come, MandrakeSoft wishes to implement a SAP type of a model.

4.4 Economic model.

4.4.1 Stock market listing.

http://www.boursorama.com/graphiques/graphique_histo.phtml?symbole=1r04477-OTC What follows is the slightly reformatted message that Jacques Le Marois sent to the community just before the company was listed on the stock market in July 2001. He chose to do so to explain why this listing was a necessary step. His explanation proved to be quite useful. Indeed, there are still detractors today who believe that this was treasonous vis-à-vis the community. Curiously, these detractors often are in France which is where MandrakeSoft started off from.

> «Founded in 1998, MandrakeSoft has often been cited in computer magazines as one of Linux's greatest "success stories". Indeed, within less than three years, the company had achieved its first and very ambitious objective, namely, conquering one of Linux's largest user bases around the world. By so doing, it had received the most prestigious rewards within this field as well as the unanimous respect of its users and developers. Once this goal had been fulfilled, the company decided to take up another ambitious challenge: to transform an increasing number of users into Mandrake Linux products and associated service buyers. MandrakeSoft has indeed become one of the Linux world leaders. This is particularly true in the United States where, according to PC

[17] «Sincerity», which is why, in this section entitled Strategy, we hesitate to remind the reader that, recruiting well-known free software partners (Cf. infra.) through complete sponsorship, offers both a competitive advantage and a stronger public image.

Data figures, the company was number one for the sale of Linux products for the first quarter of 2001. Nowadays, an increasing number of large businesses turn to its customized development services.
MandrakeSoft fantastic potential is yet to be exploited. Several development levers are going to contribute to the acceleration of the company's growth so as to meet the following two objectives: higher yields and a return to profitability:

- new infrastructure solutions are going to be available for businesses
- the number of indirect distribution channels in an international framework as well as on-line sales is going to increase
- traditional services will be more developed among the numerous businesses which use the Mandrake Linux operating system —training, technical support and consulting—
- new on-line services dedicated to Open Source information technologies will be exploited (support, forums for exchanges).

The listing of MandrakeSoft on the stock exchange aims at financing this important development program.»

All 688 480 shares were subscribed for 6,2 euros each, via the Placement and the Fixed Price Offer. The total sum amounts to 4,3 million euros. New shares represent 20.28 % of the capital after flotation (emission).

4.4.2 Mandrake in the beginning of 2002.

MandrakeSoft closed its fiscal year on September 30, 2001. The consolidated results were accepted officially by the board of directors, which met on December 12, 2001.
The 2000-2001 consolidated turnover amounts to €3.6M and shows a 18.9 % increase. However, this

30/09/01 (€M)	1999	2000	2001
Turnover	0.6	3.0	3.6
Operating results	0.1	-6.5	-13.4
net Results	0.1	-6.8	-13.6

Table 4.1 **—** Tables and commentaries taken from companynews, based on an early 2002 Mandrake release.

performance is slightly below (€0.7M) what had been planned during the subscription to the Free Market. This is due to the slowing down of retail sales "retails"[18] (73 % of the turnover) both in Europe and in the States (65 % of the turnover) which had started in July and which was even more pronounced after 9/11. On the other hand, on-line direct selling through the MandrakeStore Website (9 % of the turnover) and the Services for businesses (16 % of the turnover) reached their objective, thus validating the new commercial strategy of the group. It should be stressed that, both overhead and staff costs were cut by half between April and November.
After a first stable semester in 2000-2001 (September-April), MandrakeSoft managed to accelerate its growth during the second semester with an increase of 45 % of its consolidated turnover in comparison to the preceding fiscal year. Little by little, it also reduced its workforce.
As planned, the results show a deficit but, nonetheless, they correspond to the estimates. The first semester was strongly characterized by a lack of results in the diversification strategy towards *'e-learning* which had

[18] through mass marketing, the Fnac...

been conducted by the international management team that was recruited at the end of 2000. The heavy expenses (men / R&D / marketing), linked to this policy had mechanically penalized the company's results. As a consequence, MandrakeSoft had to re-focus around Linux by implementing drastic measures in order to reduce its expenses and to improve its margins during the second semester:

- There was a change in the company's management and no more diversification *e-learning*,
- The workforce was cut by a third and the total workforce was reduced by 57 %,
- Overhead was cut in half,
- More profitable activities were launched (e-commerce, services)

These measures have borne fruit since, while MandrakeSoft managed to accelerate its growth during the second semester, it also succeeded in cutting both its overhead and monthly staff costs in half between April and November. The latter came out at the end of 2001 0,7 ME. Although these results were already perceptible in the fiscal year which closed on September 30, 2001, it was believed that their impact on the company would probably show up only in 2001-2002.
Under these conditions, MandrakeSoft planned to reach an operating balance for the last quarter of 2001-2002 fiscal year.
In addition, MandrakeSoft wrote out a money order to the KBC Securities Bank for an increase in capital reserved in favor of financial investors or industrial partners. The objective was to accelerate the development of MandrakeSoft on the market, either through the recruitment of dedicated teams or through an external growth.
In March 2002, the number of securities is 3,410,000 for a total market valuation of 10.2 million euros.

4.4.3 Creation of an Users' club.

The users' club has been a new source of income since the end of 2001.

- *Mandrake Linux Users' Club*, with over 2000 subscribers registered and 10000 expected within a year from now. The first level of subscription costs to the club is less than €6
- *Mandrake Linux Corporate Club* For businesses which use Mandrake Linux for their daily activities, the subscription provides specific recognition and exclusive privileges (cost of the subscription: €3,000 or more).

For a few weeks now, Mandrake has asked all those who use its products to join the club. Indeed, despite laudatory media coverage, the craze for the latest Mandrake products, and, even more so, for the company's latest innovative goods and despite revenues that keep increasing, development costs as well as the costs related to the Website services which are offered to the Mandrake Linux community, will not be covered. Financial equilibrium is estimated for the end of 2002, but it might already be too late.
MandrakeSoft has forced the community to accept its responsibilities: how far can people keep enjoying free products and free quality services? The question must be solved quickly. This should be easy enough considering that subscription fees are low and that supplementary privileges are then offered. Yet, the community remains divided between those who find this quite understandable and those who believe that a company that works in the field of free software, should not follow such a mercantile path.

4.5 A story about men and women.

Mandrake's 2001 misadventure which resulted in a reduction of the workforce shows how paramount a good managerial choice is within a company. This seems particularly true in the field of free software. If there are no ethics, if people do not share the free software culture, the company is not viable anymore[19]. Nowadays, MandrakeSoft managership endeavors to recruit people who already have or are willing to acquire this culture. By being in contact with their colleagues, managers are getting "converted"...
The following information is about the managerial team and is largely taken from: http://www.mandrakesoft.com/company/about/executives It has to be completed with the "developers who come from the community" profiles which are proposed in the last part of this document.
Jacques LE MAROIS, CEO, Co-founder. With a degree from the École Normale Supérieure (a grande école for training of teachers) and a degree from the Collège des Ingénieurs(An advanced school of engineering), Jacques Le Marois is a pioneer in the field of new technologies in France. After the Internet "adventure", he became interested in the Linux market and contributed to its expansion among businesses. On October 10, 1998, he organized a huge, national Linux Party with the participation of 35 cities and several thousand visitors. Jacques Le Marois is used to leading large-scale projects for businesses. At 26, he took part in the launching of a new entity for the Danone Group. Then, in 1997, he re-organized and put a reporting system in place on the occupation of Andersen Consulting virtual offices both in Paris and in Amsterdam. He is one of MandrakeSoft co-founders.
Gaël DUVAL, Co-founder, creator of Linux-Mandrake. With a "network and documentary applications" post-graduate degree from the University of Caen, Gaël Duval is a Linux pioneer in France: he has insured the promotion of KDE on a specialized Website and created a list of diffusion "FREE" recognized by the specialists in this field. He has also participated in the translation of reference texts on Linux, he conceived the first Website in the world that offered an entire MP3 format disk on Internet at the end of 1997...
Gaël Duval not only has acute technical competence (necessary to build the first Mandrake distribution), but he also has the qualities required to create bonds around the world. This "dual profile" has been an inspiration to him and in 1998, at only 25, he created Linux-Mandrake. The basic concept consists in associating the highest-performance distribution on the market with the best graphic environment possible. Relying on his network of international contacts, he handles both the promotion and diffusion of the first versions via Internet.
Frédéric BASTOK, Technical Director, Co-founder. Frédéric Bastok is a civil engineer and has a degree from the ESTP (grande école for civil engineering). He used to be in charge of the Linux section for PC Expert Magazine. Later on, he was head of the Support department for the Opera Software company. Since the creation of MandrakeSoft, Frédéric Bastok has had a major role in the relationship with the Open Source community, as well as in defining the technological and software innovations which have been brought to the Linux-Mandrake distribution.
Marc PELTIER, Chief Operating Officer. Marc Peltier graduated from the Paris Business School (ESCP) and has an MBA. He started as a management controller at Philip Morris and became the Head of a Project for a European pilot project. Then, he joined the Cimad Conseil company (owned by IBM) as Project Director specialized in the implementation of ERP projects. He took part in the development of Service Offers. When Cimad Conseil was integrated in IBM Global Services, he was promoted to be Head of the important accounts (highly powerful private and governmental organizations) for the industry sector within the ERP Service Offer Department. In February 2000, when he joined MandrakeSoft, Marc Peltier was MandrakeSoft Services Director. He is now Chief Operating Officer.
Daniel MORALES, Vice-President, MandrakeSoft Northern America. System and Chemical engineer, Daniel Morales majored from the Monterrey Instituto Tecnologico (Mexico). He also has an MBA from the University of California (Los Angeles). His role consists in supervising and implementing new initiatives for MandrakeSoft in the United States, Canada and Latin America. Before he joined MandrakeSoft, just a few

[19] This may have been the case for Linuxcare.

months after its creation, Daniel Morales was Director for the Economic Development and Assistance Programs for Businesses in the Los Angeles area. Head of the administration, he succeeded in obtaining $125 million in loans and financial assistance programs for many local businesses. He also founded, launched and managed numerous startups within a multinational framework.
Barry COCHRANE, Sales and Services International Manager, Europe. As Sales and Services Manager, Barry Cochrane is responsible for the implementation and management of the best methods and of the optimal development plan for MandrakeSoft in Europe. Barry Cochrane has over 30 years of experience in the computer world as well as in finding technological solutions for businesses and international markets. This has led to the listing of companies such as Sybase and Pure Software on the stock market. Before he joined MandrakeSoft, Barry Cochrane was the European Manager of Numéga (Compuware). Then he launched Linuxcare in Europe. Barry Cochrane started his career in the Royal Air Force as a systems analyst, then assumed different responsibilities in Cullinuet, Honeywell and Admiral.

4.6 Interactions.

As we have said earlier in the Community section, MandrakeSoft supports many organizations and Free Software projects. Recruiting such figures through, sometimes, complete sponsorship offers both a competitive advantage and a superior public image for the company. Besides, we must say that, right now, all of the most famous developers in the field of Free Software have been contacted by businesses. There are not that many that are still available.

4.6.1 Free Software Foundation (FSF) Europe.

Project Manager(s): Richard Stallman (Founding Father of the GNU project and Free Software Foundation). Bernhard Reiter, Peter Gerwinski, Werner Koch et Georg C. F. Greve. Date of creation: Winter 2000 Description: The objectives of the FSF Europe consist of: coordinating FSF initiatives in Europe, providing the necessary infrastructure for the projects which are related to free software (in particular, the GNU project), and the implementation of a competence center especially designed for politicians and the media. MandrakeSoft contribution: On December 28, 2000, MandrakeSoft gave 2500 EUR to Free Software Foundation Europe, which is the sister organization of the American FSF. This helped to allow FSF Europe to face the legal expenses contracted in January 2001. How has the project progressed? FSF Europe has just been created. Its activities will evolve at a sustained pace in Germany, France, Sweden and Italy throughout the first semester of 2001. Other countries such as England, Belgium or the Netherlands are expected to follow suit. For extra information: http://www.fsfeurope.org/

4.6.2 GNOME Foundation.

Date of creation: 2000 Description: The GNOME Foundation is a non-profit making association which is meant to help its homonymous project advance. The GNOME project gave birth to a completely free and user-friendly graphic environment, as well as [framework applications] on Linux and other operating systems which are based on Unix. GNOME is part of the GNU project. The GNOME Foundation will later provide the financial, legal and organizational support necessary to the project. It will also help determine its philosophy and its objectives. MandrakeSoft contribution: MandrakeSoft joined the GNOME Foundation Advisory Committee on December 14, 2000. The Linux-Mandrake distribution included the GNOME environment right from the beginning and has become Mandrake users' favorite graphic environment. MandrakeSoft will later on sponsor GNOME developers financially, so as to help them focus on the development of GNOME projects. For extra information: http://www.gnome.org/

4.6.3 KDE League.

Date of creation: November 15, 2000 Description: The KDE League was created thanks to an alliance between very important industrial groups and KDE developers. It was meant to favor the promotion, distribution and development of KDE. Later on, the League will focus on promoting the utilization of KDE, which is the very advanced and alternative of Open Source bureau, on PCs, work stations and mobile objects (whether for businesses or private individuals. The League will also endeavor to incite third party developers to work for KDE. MandrakeSoft contribution: As a founding member of the KDE League, MandrakeSoft joined with other major actors of the computer sector to provide KDE with necessary financial, moral and advertising support. For extra information: http://www.kdeleague.org/

4.6.4 KDE.

Project Manager(s): David Faure, Mosfet (Daniel M. Duley) Date of creation: October 1996 Description: KDE (K Desktop Environment) is a powerful Open Source graphic environment for Unix work stations. It combines user-friendliness, very modern functionalities and amazing design to Unix technological superiority. KDE definitely is an Open Source Internet project. KDE has developed a high quality Unix [development framework], which allows fast and efficient application creations. KDE is the result of the work of hundreds of developers from over 30 different countries. MandrakeSoft contribution: MandrakeSoft recruited the following three developers, David Faure, Laurent Montel and Daniel M. Duley (Mosfet) so that they would devote themselves to the development of KDE-2. MandrakeSoft also hosts the Webcvs website which is dedicated to KDE. How is the project progressing? On January 31, 2001, the KDE team announced that KDE 2.1-beta2 was available. It consists of a powerful, modular graphic environment which has been optimized for Internet. KDE 2.1 is the second major version of the series number 2. For extra information: http://www.kde.org/

4.6.5 Plex86.

Project Manager(s): Kevin Lawton Date of creation: August 1999 Description: The objective of the Plex86 project is to create an evolutionary Open Source virtualization application. This will allow PC and work station users to work on several operating systems simultaneously. Plex86 will have most of the operating system and application functioning as naturally as possible. The rest will be emulated by the PC virtualization system. MandrakeSoft contribution: MandrakeSoft recruited Kevin Lawton, Plex86 Project Manager, during the summer 2000. How is the project progressing? The first user application was planned for the summer 2001 For extra information: http://www.plex86.org/

4.6.6 Bastille Linux.

Project Manager(s): Jay Beale, Yoann Vandoorselaere Date of creation: December 1999 Description: The Bastille Hardening System project aims at making Linux operating system more secure. Right now, it works with Red Hat and Mandrake distributions. The development team wishes to provide the safest system possible, without it being overly constraining for the user. The project is lead by Jon Lasser (chief coordinator) and Jay Beale (chief developer). A large number of developers, beta-testers and designers are also involved in the project. Those who designed Bastille Linux had several objectives in mind: exhaustiveness, education and community. MandrakeSoft contribution: MandrakeSoft recruited Jay Beale (chief developer) on November 14, 2000, to manage the team dedicated to security and which continued to grow. How is the project progressing? Bastille Linux is continuing to develop. It now has a new user interface and updates for firewall functionalities, XINETD and DNS optimum. For extra information: http://www.bastille-linux.org/

4.6.7 Prelude.

Project Manager (s): Yoann Vandoorselaere Date of creation: 1998 Description: Prelude is a network intrusion Detection System. It consists of both Prelude and Prelude Report programs. The first program makes it possible to capture packets and analyzes them via plug-ins. The second program reports attacks to the user in a readable manner. This allows him/her to get reports on another machine. Prelude's other major characteristics include an IP defragmentation pile and facilities provided to plug-ins in order to guarantee stability. MandrakeSoft contribution: Yoann Vandoorselaere, project manager, was recruited by MandrakeSoft in February 1999 to provide Prelude with the help it needed. How is the project progressing? Prelude version 0.4.1 has just been created. For extra information: http://www.linux-mandrake.com/prelude/

4.6.8 RpmLint.

Project Manager (s): Frédéric Lepied Date of creation: October 2000 Description: RpmLint is used to check errors which are common when RPM packages are created. This way, source and binary packages can be checked. Mandrakesoft contribution: In December 1999, MandrakeSoft recruited Frédéric Lepied, to be their project manager. It provides him with all the support he needs to develop the project. How is the project progressing? RpmLint has realized its 0.29 version.

4.6.9 Urpmi.

Project Manager (s): François Pons, Pascal Rigaux (Pixel) Date of creation: May 1999 Description: Urpmi allows an authorized user to make packages from several media sources. Its main functionalities are: automatic management of dependences and search for expressions in the list of files of all the packs that are managed. MandrakeSoft contribution: Pascal Rigaux, the creator of the project, was recruited in July 1999 by MandrakeSoft. The company has never ceased providing him with the support he needs to develop the project. How is the project progressing? Urpmi has realized its 1.5 version.

4.6.10 PHP-Nuke.

Project Manager (s): Francisco Burzi Date of creation: summer 2000 Description: PHP-Nuke is a turnkey-application similar to Slashdot. It allows anybody to easily create new functionalities for the Website according to a community mode. The objective of PHP-Nuke is to provide an automated site which allows the diffusion of news and articles. The site even offers a user account system. Each user can publish commentaries concerning the articles. Available in 23 languages, PHP-Nuke has already been downloaded by more than 140,000 users within six months and hundreds of sites use it around the world. PHP-Nuke is entirely written in PHP. It requires a Apache Web server, the PHP language and a MySQL database. MandrakeSoft contribution: When he joined the MandrakeSoft team, Francisco Burzi, PHP-Nuke project Manager, received the financial support he needed. Moreover, he was provided with total technical support: Website hosting, mirror sites for downloading. He also benefited from the Mandrake communication network to make PHP-Nuke known among users worldwide.. How is the project progressing? The most recent version of PHP-Nuke is the 4.4 version. It became available in February 2001. For extra information: http://www.phpnuke.org/

4.6.11 Development of Kernel Linux.

Project Manager (s): Linus Torvalds Date of creation: September 1991 Description: The kernel is Linux operating system's fundamental component. Linus Torvalds programmed it from scratch with the help of a not so

well structured team who worked via the Internet. It has all the characteristics of a real, modern operating system. The kernel is distributed under the General Public License. MandrakeSoft contribution: one of the major developers of the Kernel is Jeff Garzik. He was recruited by MandrakeSoft in October 1999. MandrakeSoft provides Jeff Garzik with the financial and technical help he needs so that he can keep on contributing to the kernel for the good of the Linux community. Philip Rumpf and Juan Quintela were also recruited to contribute to the development of the kernel. How is the project progressing? The most recent version of the kernel is the 2.4. version For extra information: http://www.kernel.org/

The Free Software Service Companies: the exemple of Makina Corpus.

AYMERIC POULAIN-MAUBANT,
Jipo Interactive, Aymeric.PoulainMaubant@jipo.com.

5.1 Makina Corpus

- people who were interviewed: Jean-Pierre Oliva (general manager), Philippe Julien, Mose
- Reference site:

  http://www.makina-corpus.com/

Makina Corpus is a SS2L (Free Software Service Company) which provides services and Internet and telecom applications. It is established in France, in the United Kingdom and in Spain.

It was created by a telecom engineer who had a lot of experience in counseling. He has affinities with Free Software, mostly as a user, rather than as a developer. He still is general manager and runs the main strategic orientations.

5.1.1 Activity

Makina Corpus works around 4 poles:

- *Development* of reliable and powerful Web solutions and of application development in areas which range from telecommunication systems to mobile telephony applications, from Intranets to e-business sites, from mobile Internet to m-business sites,
- Strategic, technological *advice* to project owners and project management, assessment of people's needs, audits and specification of terms and conditions...
- *Training* on free software, such as, for example, that intended for Internet access supplier call centers, which have to deal with more and more Linux users. This training is based on continuous monitoring (which is itself, an available service),
- *Services*, installation and administration the total number of computers, migration, distance administration, information management and user assistance...

In addition to the fact that Makina targets service providers such as Internet (access suppliers, hosting...) and telecommunication service providers (mainly the operators), Makina also targets *Web agencies* likely to use management software of the contents of Website developed by the latter. Besides, the number of customer types may increase through Makina consultants who have been sent on-site to assist the customer and who detect (with the customer's agreement) the needs of this customer's own clients.

Free software is, above all else, a choice of tools and methods. Makina Corpus has made this choice because it allows permanent openness towards new technologies, guarantees independent technical choices and makes it possible to concentrate on what matters most: bringing the best adapted answer to the customer's needs. This is why, even if Makina provides what some people might call traditional services, such as site installation, heterogeneous integration maintenance, preexisting free software adaptation and made-to-order development, tasks which are usually associated with computer specialists, its value also lies in counseling and assistance services (such as, for example, intermediation between independent firms and developers). Such an approach allows better understanding between the sometimes chaotic world of free software and that of business sector.

Two types of software belong to the Makina Corpus "portfolio": those which are produced on the outside, like Zope (and others which are continuously tested, then added to Makina competence), and those which are produced in-house. The latter clearly derive from Raymond [1998a] principles: "a developer works all the more easily on a tool which he/she personally needs". Note that these software programs are progressively improved as they are implemented among Makina customers. We will now give more detail on some of them.

Zope

Zope is an application and content server, supported by a significant community. It makes it possible, in particular, to allow non-technical individuals to manage complex sites.

Zope is not developed by Makina Corpus and, for the time being, no member of this organization is taking part in its development, at least, in a sustained way.

CCMS

The Collaborative Content System Management is a Website management tool through an interface. It is written in php, based on an xml storage file, with an ldap, mysql modular authentication or textual file. Its characteristics are designed to provide access, via an ergonomic user interface, to all the elements which constitute an evolved, multi-field and multilingual Website: template systems which can be customized, improved textual stylesand variable management according to the context. This tool was conceived to make it easier for the various participants of a site to work together: design, contents and referencing can be managed directly.

Its main author is Mose.

CCMS is not really based on proprietary software, although there are some with similar objectives. On the other hand, it is quite similar to other free software tools, like *cWriter* (see infra.), *spip* or *dacode*[1], tools that emerged at the same time that Mose's first drafts did.

As for cWRITER (one can sense the author's style), its originality comes from combining different traditional elements. The way the interface is organized creates a context which is specific to this application. These are CCMS's strong points: infinite historization, use of a *a rich text*[2]. This is also the case for *wikiwiki*[3] or spip. However, these can be defined by the user. There is an extremely wide range of parameters to influence the site's visualization mechanics...

CCMS principle stems from the daily practice of creating sites. This has been going on since the beginning of the Web. The solutions that are implemented correspond to immediate production needs. CCMS's predecessor was developed a few months before the foundation of Makina Corpus, in February 2001. It was intended, at the time, for two distinct uses: on one hand, the management of the claranet site [4] and, on the other hand, that of the Cardionews [Mose]site (an association of cardiologists). The claranet site was never finalized, but the cardionews site was. It is now operated by keo.net [5]. The first sites that were ordered from Makina Corpus had a management template. It became more and more refined as the sites evolved since it was confronted with various constraints (*frames* or not, need for personalized forms, atypical

[1] Respectively http://savannah.gnu.org/projects/cwriter, http://www.spip.org/ and http://www.dacode.org/

[2] the user does not have to know the codes HTML for bold types, for example, all he/she has to do is type the word in *** bold *** with stars. This does not break the rhythm and it does translate the notion well in a visual way

[3] http://c2.com/cgi/wiki?WelcomeVisitors

[4] http://www.claranet.fr/ Internet service provider where Mose was before it joined Makina Corpus. See section RH

[5] That is to say that the service which is associated with the software is accomplished by a third party.

models, different sites that had the same model, etc).

packaging in view of its diffusion under the Free mode started in September, to be completed in December 2001. At the beginning of January 2002, going on-line provided it with the opportunity to make some positive contacts and Makina Corpus is confident about the appropriation phase that is currently under way. (see also under Community section).

cWriter

cWriter is a tool in php and mysql which proposes an on-line environment for editorial collaboration for group of 10 to 100 people [6]. Functionalities were added to the editorial system so as to improve editorial collaboration: collaborator/writer system , shared planning, directory, contact forms, file downloading, bank of links, access to consultation statistics, printable version, integral version of each document, a locking system, inclusion of comments in the body of a document, etc.

Its main author is Mose.

It is more or less based on proprietary software (Lotus) or semi-proprietary software (BSCW): more with regard to objectives and uses that are targeted and less with regard to the interface, specific functionalities and the approach that was chosen. It was developed following attempts at collaborative work on these proprietary platforms which entailed heavy management costs and a non-capitalizable specific expertise. At the time (mid-nineties), only BSCW could lend itself to a context of collaborative work, but the complexity of its interface shattered the appropriation rights from the moment the first tests were done. After that, development phases were spaced out in time, over years, according to the communities' various experimentation [7]. There were no preliminary strategic plans except that they wanted to encourage better reliable and fluid collaboration.

Its originality comes from the combination of some of its functionalities which, in addition, are quite traditional. The result is that this product is unique. Nothing is really original as far as most of its functions are concerned, but the way they were put together shows that their use was observed in-depth. Some details, however, are less common. For example, each typing-in stage is saved allowing infinite access to the background of the entire process. The interest of cWriter lies in the concentration on functionalities which focus on the formalization of a written document. But this can be used differently and some documents written under cWriter could be described by those who use them as forums of debate or polling systems.

RC's first draft was called 'el communiante'. It was a fast, collaborative editorial tool which was produced as soon as php3 was available on the market during the summer 98. The objective was to write the technical follow-up file of an event that Wanadoo had organized. For the occasion, Wanadoo had hired the Associated Net surfers: Mose was one of the satellites used. Domenica Large, consultant in Distance Training at the time, worked a lot on the functional aspect and implementation of test stations. But it was still too early and Mose's sporadic availability stopped him from meeting deadlines. It was rewritten several times, was called "orc", "arc", then finally "rc".

It was developed experimentally for the university of Clermont-Ferrand Amberlab [Mose] in community circles (Cape-Corsican then Radio-Phare, but also Acuarel [8]), and in-house, at Claranet, for the management

[6] Note that 80 % of this monograph material was built from an especially dedicated RC. The author had asked his/her own questions on it and various contributors answered and even exchanged views among themselves on a document that was being built.

[7] the term "community" should be understood as a group of people who have affi nities.

[8] See http://www.radiophare.net/. The author of this monograph belonged to that small Acuarel community which included sociologists, economists, geographers, computer engineers, students and researchers, veteran Net surfers who had met on the Internet or in former lives! In May 1998, they wanted to discuss Manuel Castells' fi rst volume on Networks in Society (Castells [1998]). Acuarel was an acronym for "Architect, Concièrge (janitor), author, Rédacteur (writer), Éditeur (editor), Lecteur (reader)", which were the

of knowledge bases.

In November 2000, the source code was offered to the community and diffused on Sourceforge, under a GPL license. Since then, it has been loaded approximately times, which is negligible, on one hand, and yet, considerable (from the point of view of the author who receives as many comments which can be used to improve the software). It is necessary, however, to note that Mose thinks that he received greater support from his closest relationships, who had gone beyond the threshold of appropriation, than from the enthusiasm of the free community. With hindsight, he explains that this was due to the choices that had to be made with regard to development organization.

Indeed, to be free (that is, free from the influence of its author [9]). So, to be free, a software must initially have *script* an effective installation, which will make it possible for other developers to test it, even if nothing works. In the case of cWriter, the installation *script* was produced tardily, and the appropriation of this tool by the Free community will come late. To tell the truth, it has barely begun.

5.1.2 Competitors

Zope can indeed be compared with both proprietary and free software programs (Cold Fusion, MidGard, Enhydra, Interwoven, even Broadvision and Vignette). It has strong points and weak points, in particular those related to Python, but the software ecosystem in this type of applicative is still young and leaves enough room for everybody. However, its visibility is not perfect and it does not have enough hosts.

CCMS has few competitors in its category. Taking the context into account (an application to make Web sites based on the Web), competitors (Web agencies, for example) do not advertise much because it is often easy to take others source codes (in particular, when functionalities are in Javascript). This software is thus either protected in extranets or has its main components from the server point of view (this is close to being a proprietary software).

Perhaps CCMS lacks stability (new functionalities are added daily) and it does not have that many installations either.

Another free software, eevote/Glasnost [10], proposed by Easter eggs and based on Spip and python, seems to be working towards the same ends as the RC or the CCMS (even these two are sometimes similar). However, it concentrates on the vote tool before anything else. Mose estimates that, in this case, neither a rapprochement nor a fusion are envisaged. There is no conflict either, but rather a natural reflex of negation of similarities so as to avoid this potential conflict: it "a kind of derivation in the development axis can be felt on both sides, actually. But the Free software system tolerates (and favors) people much better than the commercial approach for which the lever effect is paramount (and acts as a powerful stimulus indeed, since there is not much to lean on"

5.2 The place of the Community

The Community, in the strict sense of the term, is not yet very present around Makina Corpus. But the developers of Makina come from the Community, or from a feeling of community, or from small communities. We

roles endorsed by RC Users.

[9] This is certainly true in the case of a very small group of authors, sometimes, only one author, who, at the beginning, is happy with a small development environment. This is definitely less true when authors are very numerous (several dozen or hundred). The typical case is that of the Debian distribution which absorbs its authors instead of expanding itself.

[10] http://www.entrouvert.org/rubrics/2

will deal with this point in the Human Resources section. They activate their community networks, frequent communities (Atica...), look for synergies and customers there.

Because Makina Corpus develops free software, the Community is a significant source of input: the latter must be able to test, comment on, criticize, adapt, divert this software so that the latter can develop[11]. Moreover, prescribers among Makina Corpus customers are increasingly people who have witnessed the very beginning of Internet for the general public and Linux (first half of the 90s), that is to say, they come from or are close to the Community.

Makina Corpus is starting to propose a Website especially dedicated to inquisitive developers and users Mose [2002], on which the software that has been developed is placed, as well as examples on how to use free software in telecommunication contexts. The latter provide internet or mobility services and these are major axes for Makina.

This having been said, it does not mean that there is a plan to develop services sourceforge "style". Both means and resources are lacking and, especially, because these services already exist. And even more than that: services like Savannah or Tuxfamily [12] already host communities through which exchanges can be fruitful. " [These are places which allow] to confront the product with the reality and, in an eugenic goal of natural standardization, even modify it so that it can be adapted to what is done." For example, to be accepted on savannah, there are conditions that must be met. This was not the case when RC was proposed there. Mose had to work on it again so that it would meet the (Savannah) standards: more detail on the presentation of the license, absence of gif format images. RC was then accepted. For the community, these rules are a token of quality and a guarantee not to be in unknown territory all the time. In the same way, the CCMS was refused whereas RC was accepted, for file header problems whereas Mose had paid attention to this (following the first attempt for RC to be accepted). Therefore, for a developer, nothing can be taken for granted and, yet, this is not even about the rules to follow to include a product in a GNU project [13]. Besides, it is interesting to note that the rules which structure the form of the software also structure the community which adopts them.

Perhaps the site suggested by Makina will also find its community of users.....We will meet again next year.

5.3 Promoting Free Software programs

5.3.1 The necessary promotion

At Makina Corpus, the author himself (and the developers) of a free software program ensures its promotion. He/She knows the niches —of the community— where his/her work will be backed by *the ' peer review'*. It is much preferable to have this done before mass diffusion starts. Nevertheless, developers rarely know how to talk about what is too close to them. When the time comes to promote the image of the company and the value of its elements, it is better to call upon somebody who has a stronger marketing profile. Therefore, Makina Corpus called upon some of Jipo's experts to actually test the context of communication around free software. It then drew up a battle plan involving numerous tactical axes. On the other hand, right now, there is still one single person whose sole role would be to apply Jipo's recommendations [14].

All the people involved in Makina Corpus who find themselves in an appropriate context (customer,

[11] grow / mature and get diffused.
[12] Resp. http://savannah.org/ and http://www.tuxfamily.org/.
[13] In this case, the structure of the documentation, the way the installation is handled, the writing of the code are strongly controlled.
[14] in February 2002. Things are moving quickly according to the profi les recruited.

conference, newsgroup...) must be able to ensure the promotion of the Makina structure (software, services and know-how). The customer him/herself must become an ambassador of the tool that was developed for him/her. Of course, when the development proves to be a good, this is done in a natural way. Lastly, taking part in workshops with solution researchers must be part of the promotion creed (Atica, decision-maker seminars...). This work can be undertaken by two individuals at the same time (a developer and a non-developer, in marketing or not) so as to answer the various types of interlocutors one meets in this kind of event.

The existence of quality support also helps promote a product. This is particularly relevant for Makina since support is one of its commercial axes. An instruction manual is always provided, but to develop one of Makina software programs in a particular context requires some advice: this is usually done by the developer him/herself. He/she goes and meets (and listens to) the customer. As a consequence, this allows him/her to improve the new versions of his/her software thanks to his/her customer's specific remarks and needs.

However, this does not mean that the "spirit" of Free software has been tarnished. The software comes with whatever it takes a well-informed and qualified user to ensure his/her own support. Of course, in this case, there is no immediate return in terms of sales turnover (the software is free and no support was sold). On the other hand, this category of user will tend to appropriate the software, to appreciate it and to recommend it, in particular to less well-informed users. He/she will then recommend the support provided by Makina. This approach works over the medium term (however long it takes the customer-ambassador to appropriate the software).

It should be underlined that the promotion of Free software is less and less necessary. For a few years now, in France, many associations (to take this one example) have informed companies, communities, the administration, the media and the general public on what is at stake with Free software. It is quite common to have customers spontaneously ask to produce free software (or in the spirit of Free software), to ask for moderate costs (let us say, fair costs) and for reliability and security. The administration, which is one of Makina Corpus targets, follows governmental directives which are less and less unofficial. Positive effects can already be felt (significant migration contracts from the world of proprietary software to that of free software).

Customers do ask for free software, but the license is not a fundamental element in this decision-making process. It is true that the word "license" covers a very different economic reality depending on whether one is considering the worlds of proprietary or free software. In the case of large companies, the price of the license, even if it is high, remains marginal compared to a specialist consultant. These large companies are a lot more attracted by the concept of "support by a broad community" since this guarantees them with the continuation of its software (and data). A small structure (public or local authorities, small and medium-sized businesses) remains sensitive to moderate costs and generous licenses.

5.3.2 Licenses

The GPL chose the FSF for several reasons. On the one hand, a guaranteed non-mercenary approach is a good sales point for customers who pay development services (when they ask to for integration or for customized development). Since this license guarantees that final improvements of the product will remain free, there is no danger of captivity by the editor.

This choice was also justified because, in the Free community, this license signals proof of integrity, which can be regarded as a factor supporting appropriation.

5.4 Strategy

Makina Corpus did not chose a 100 % Free strategy. There is no reason to refuse customers who come from the proprietary world or who want to have proprietary products (this is something that is quasi-obligatory in the world of telecommunications). Some of the services that Makina provides, such as the (tele-)maintenance of computer systems, are occur in strongly heterogeneous environments.

On the other hand, original skills are clearly articulated around Free software. As software producers, Makina has definitely chosen the Free mode [15]. Makina wants to stress services, attract and keep customers thanks to quality services rather than by locking licenses and having ceaseless updates... Makina developers also profit from the feedback of contributors and can thus improve their products. This, in turn, has accelerated *the time to market* Makina solutions [16].

As users of existing free software programs, Makina seeks the specific assets of Free software (quality of the developments which are available, greater stability, open-mindedness with regards to criticism and thus to improvements, the benefits it can gain from evolution brought by the community). Makina does not want to be subjected to the diktats of software suppliers. It wants to be able to adapt existing software (this is necessary for telecommunication customers who are always in demand for new and quickly developed applications).

Makina encourages its employees to contribute to free projects which are external to the company, for all the good reasons one can imagine: it is a means of following the evolution of this software, it makes it possible to propose new ideas and have them validated, it makes it possible to make Makina known by developers, by customers, it is a guarantee of competence for the customers, even sometimes an express request on the behalf of the customers.

Right now, the company wants to create a software portfolio in which Makina can show off its skills (counseling and training, specific developments and integration, etc.). Consultants, who always listen to customers, spot software programs that can be directly useful, or will be, once they have been adapted and they test them. Of course, this efficient approach is preferred to that of having to develop *from scratch* .

This approach implies that all consultants must be permanently on the lookout (to find and seek critical testers). Consequently, communication costs drop because the relational network of each one of them extends very quickly. This is a virtuous circle because, the broader this network is, the more third parties adopt software solutions developed by Makina. In turn, the number of new prospective customers who are invited by third party-first-testers grows. These potential customers are less well-informed technically or have less time and, as a consequence, are fond of getting advice, assistance and support services.

Besides, it is beneficial to expose oneself to criticism, when both quality (and reactivity) are there.

Finally, let us mention the last advantage of this strategy towards Free software: to take part in a free project is, sometimes, to benefit from the proximity of experts and get trained in a particular field.

5.5 Economic model

Makina Corpus positions itself as both a consulting group and SSII, not as a software editor. Therefore, its economic model is, quite simply, that of a service company. No one can say that the Free software mode

[15] There is no way that Makina plans on developing proprietary software while, at the same time, it praises the merits of Free software.

[16] Makina is a important free software user but it also wants to contribute to the community.

adds a new dimension here, especially when one knows that Makina also works in proprietary environments. All types of customers are targeted but, for historical reasons, the telecommunication sector, Internet service providers, and either public or private authorities, play a very significant role today.

Income comes from counseling and services (analyses of the customer needs, personalization and customized development, maintenance, training, etc.). It is also possible to provide customers with our consultants' competence (this is usually done through line supports) [17].

It is clear that it will take more time for income that derive from counseling to be completed than those which derive from development and service provision, since these correspond to immediate needs.

Makina Corpus is not dedicated to one software only, even if, sometimes, there is an arbitrary bias in favor of Zope. As a whole, Makina can be considered as a non-specialist group, yet, it does specialize in Website applications, and particularly in collaborative enterprises. It also deals with development, under Zope, with Website content management systems, mobility (GPRS, PDA, *wireless LAN* ...) and geographical information systems. It is dedicated to sectors like telecommunication, the car industry or public authorities.

Since Makina sells services, the installation of the software is integrated into these services and is not handled by third parties. It deals directly with its customers, without either intermediaries or retailers.

Makina invested most in its employees' competence and strengthened their knowledge on basic tools and created a solid development framework. The company is still quite recent and it has mainly invested its time in positioning itself in a decisive way. This strategy is starting to bear fruit.

After one year of activity, Makina Corpus reached a balance. Prospects for growth are excellent. While the company was in its preparation phase, it had already won over and satisfied many prestigious customers.

The main difficulty it has had to deal with is the reticence some people have had with respect to free software.

Note that Makina Corpus does not respond to bids that do not emanate from an identified and known person.

5.6 A story of men and women

One individual coordinates commercial functions and customer relations. He/she has little experience in the world of free software, since he/she often comes from the world of proprietary software. Another person coordinates the development team. He/she is a free software activist. However, operational management is a lot more subtle than that. It has a hierarchy which shifts with the wind, as we will see.

Under the leadership of the general manager, three great functions are to be found:

1. *Operational* : development teams, missions and projects (cf. mission infra directors)
2. *Support* : marketing, commercial (cf. infra) and office automation (the latter can also be seen as an operational function, as a service)

[17] Their competence must be thorough and checked at the time of recruitment. There must be an in-depth and continuous discussion with the consultant: his/her curiosity, participation in external Free projects, implication in innovating customer-projects.

3. *Management* : administrative, financial and staff management

5.6.1 Organization

Makina Corpus is organized according to a fluctuating hierarchy. At a given time, on a given project, the external observer will conclude that there is a certain degree of hierarchy among individuals. Yet, the minute these given elements change, the hierarchy itself will have changed. The same person can indeed belong to a team and yet, be someone else's supervisor, according to the project or competence involved. Nonetheless, some stable points remain. The general manager directs the strategy and the administrative structure (for example, holiday management).

Each hierarchical person in charge has several attributions:

- organize work, entrust colleagues with tasks,
- require reports, documents and have access to the entire production,
- manage time-tables, approve dates of vacation,
- *reports* and analyze results for the general manager.

Makina Corpus functions in a project mode. The hierarchy is not permanent but depends on the project in progress:

- a sales manager for commercial activities,
- a mission director in counseling missions which require it,
- a production manager for development projects or during intermissions

In order to maintain a community spirit within Makina Corpus, it is primordial that the hierarchy remain light and flexible. This organization tries to avoid a burdensome hierarchy. It gives everybody enough freedom so that they can all test themselves by assuming various roles. This approach definitely attracts those who apply to Makina Corpus.

Production Manager

His/her role consists of:

- along with the marketing person who is responsible for a prospective customer, the production manager must clearly determine what is needed (agreement on the object of the service, its extent, deliveries, deadlines, duration, resources),
- name the developers who are most capable of carrying this project through,
- define and make sure the time-table will be respected, make sure development teams also meet their own deadlines,
- guarantee work quality for developers and support them,
- validate technical orientations,

- train and inform the commercial team on all the tools, products, developments and technical skills which are developed by Makina Corpus.

This role also includes a hierarchical responsibility over the teams that are managed.

The production manager must enter into a dialogue with the entire production team and accept some of the technical choices taken at the company level(for example, Zope competence development). He/she must also be the technical person marketing clients come to see to talk about time estimates, technical information, etc.

On particularly long or difficult development projects, the production manager names a head of project among the developers. The latter then becomes the person that marketing people come to see.

Long missions (line support in particular) are managed by mission directors.

Mission Directors

A mission director is appointed for every counseling mission and intervenes at three levels: before, during and after the mission. He is the mediator between the customer and the consultant, between the consultant and Makina Corpus; he is their contact. He works in liaison with the marketing person, the consultant, the administration and the customer.

He intervenes during all the different stages of the realization of the mission:

- he negotiates, with the marketing person, the details of the mission with the customer,
- he prepares the consultant who is in charge of the mission,
- he follows the realization of the mission and makes sure it will be properly carried out,
- he constantly checks if the customer is satisfied and keeps in touch with him/her.

The mission director reports to the general management and makes sure everything goes smoothly and that quality requirements are met. He accounts for the mission at semi-monthly meetings to the general management. He is the consultant's hierarchical executive throughout the mission and until the period of capitalization.

The mission director's main quality must be his/her competence in the way he/she manages customer relations. This is something that is appreciated, but not necessary. Besides, note that mission directors can very well belong to other companies with which Makina Corpus works, if common projects and the profiles which are available make it worthwhile. In this case, the external contributor will be trained so as to become mission director for Makina Corpus if this was not so before.

Commercial Director

Responsibilities:

- manage prescribers,
- validate marketing people's canvassing axes as well as their evolution,

- manage their priorities,
- distribute commercial in-flows between the different consultants (except when dealing with direct canvassing),
- make sure the customer is satisfied.

The executive in charge of commercial activities also closely manages two consultants who have networks and/or address books that need to be exploited in a comprehensive way. This particular, isolated type of relationship can be activated again according to the profile of such or such a new recruit. This constitutes a small, stable commercial core. Whenever needed, other consultants can join in and contribute to it.

This stable commercial core (within a flexible hierarchy) is also the guarantee that the entire commercial relations Makina Corpus has with the outside world will be filtered. For their peace of mind, no developer can be directly contacted by a new customer.

Everybody is highly encouraged to take part in commercial activities, think of new opportunities and use his/her own knowledge as much as possible. It is necessary to associate the sales manager with this approach in order to maintain a vigorous, coherent actions (one single customer should not be contacted by three different people from Makina).

In short, if a new customer contacts MC, he/she must be directed towards the commercial team. If another employee is in contact with a prospective customer on his/her own initiative, he/she can deal with the client him/herself.

Marketing Director

His/Her role consists in:

- implementing the company's marketing policy,
- realizing the various operational actions,
- continuously presenting actions and results to the general manager.

Marketing activities serve the interests of marketing people and the company. More generally, they intervene in the name of the entire company and coordinate its activities. The marketing director provides tools, actions to take for customers and for known or unknown prospective customers on behalf of the entire company. He/she is answerable to the general manager but, since the latter is often very busy, he/she is currently linked to the sales manager.

5.6.2 Who are they?

There was no community which gave rise precisely to the constitution of Makina Corpus. Makina's first employee used to work with the general manager of another company and had built a profile close to that of today's Makina Corpus consultant. At the time, they thought of building a new structure which would be organized around development, counseling and services. Mose's particular profile was rather quickly attracted by this, and by capillarity [18], a good half of the Makina's personnel come from Claranet, just like Mose.

[18] co-optation is one recruiting method among others.

This group of people definitely constitutes the structural core of Makina Corpus. In part , it provides this special mindset and work environment. Where they used to work before, they did not have much time to devote themselves to free software and we must say that coming to Makina was a great relief to them from this point of view. They are used to working together [19], they respect each other and appreciate the fact that they can still work together.

Note that this group of individuals [20] was made up of developers as well as users or, of users /developers, who had different university backgrounds. In this group, we find the same profiles as those found among customers (whether experienced and well-informed or not, whether technicians or not). Besides, Makina personnel observe that collaboration around the development of free software is seldom done among groups of people, but rather among individuals who join together and who often do not come from the same community.

This "ever changing" type of organization results from the fact that it is informal and that its initial core was never consolidated at the beginning. A core made of several individuals who, according to the project, feel closer to such or such a person. The common creed remains the Free mode[21].

Such cores are fragile: one charismatic individual might be enough to be the core of a developing community, but beyond ten people, the core can easily burst. Most of the successful Free software industries are based on a very restricted group of people and sometimes just one single individual who permutes. These observations led to the current organization of Makina Corpus. On the one hand, it must manage an internal fermentation of ideas [22], on the other hand, it must deal with external coherence.

The company has evolved a lot within one year. From a non-hierarchical group it has structured and organized itself with pre-established flows (how to handle demands, how to establish a relationship with customers, how to follow time-tables...). There is a better organized structure certainly, but also more significantly, weekly meetings on a human scale can no longer be held anymore (especially when some employees cannot be there because they are working with a customer). As a consequence, it is necessary to invent new means to continue to organize the structure while having everyone participate. Makina should be able to do this without too much difficulty since it recommends such software tools itself and since it was based on a core of individuals who appreciate each other’s company.

Does Makina grant enough autonomy to its employees so that they can keep on developing various kinds of free software? Part of the answer lies in the fact that Makina Corpus tools are free software programs. Most of the employee’s time is spent developing them. When an employee wishes to devote him/herself to a free software which is not used at Makina Corpus, the latter tries to integrate it into its standard tools portfolio and to make it one of the company’s strong points. As one consultant put it: "For the time being, employees have a freedom which is proportional to their audacity which is itself tempered by their sense of responsibility and daily self-control among peers.". No specific clause mentions anything about this in the work contract.

It is clear that the "Free mindset" has rubbed off on the entire activities of the company. In its daily management, instant collaboration, the stress put on transparency, the concern for redundancy, the desire to share, all these technical aspects have rubbed off on the human aspect too.

[19] and to using the collaborative working tools they developed themselves.

[20] two geographical pre-communities can even be identifi ed in Marseilles and in Brittany

[21] not only circumscribed to software but as an alternative way of thinking

[22] "periods of *speed* initiated by some individuals followed by periods when others cut themselves off (concentrate on what they are doing only)"

5.6.3 Recruitment

Recruitment is carried out by co-optation (this reinforces the initial core), via Internet sites (in particular those of the Free communities) and more classically via the APEC. Another element should be taken into account: the numerous and spontaneous recruitment requests we receive. It is not easy to choose between such or such a profile since the construction of a coherent, initial core will be the source of the success of this company during the first years of its existence.

This is what Makina responds to the question "what can be done to attract and retain developers who belong to neither the community nor to companies?":

"In our job, specialized technicians work more for projects than for companies [23] The cycle of employment is precarious and, yet, they are not afraid of losing their job. Managing *a turn over* should not be a problem, and the Free community, by providing some points of reference based on shared values, allows them to disregard eclecticism. At any rate, a developer is invaluable only if he/she is motivated, regardless his/her level of expertise. The spirit of competition that surrounds free software undoubtedly contributes to this motivation."

5.7 Interactions

For the time being, Makina Corpus is not a member of any particular associations, but that should change quickly (however, some consultants are, on a purely personal basis). Once someone is involved in free software, he/she often becomes a militant in order to defend him/herself vis-à-vis extremely well organized lobbies.

As a whole, employees take part in many networks (they are sometimes the founders and organizers of these networks). From the time they used to work for Makina Corpus, they have kept personal contacts with many association decision-makers who move in free software circles – while at the same time, the members of these associations are not fully aware of the existence of Makina Corpus. On the other hand, other free software companies quickly became aware of this and made contacts. Makina Corpus favorite branches of industry are particularly motivating for the teams of developers.

Makina has recently begun to work with other Free software-oriented companies since it now offers large-scale and original services. The most important thing is to recognize both the competence and characteristics of each structure. Computer engineering services cover a very vast field of competence. It seems neither possible nor desirable to be relevant in all domains for one single structure. That is why, in order to provide customers - whose requests cover a vast field of expertise - with a complete answer, co-contracting and subcontracting partnerships seem to be a viable solution.

Here, the Free software mindset is clearly an advantage. Owing to the fact that there is no need to protect one’s sources when collaborating with a ’competitor’, there seems to be an opening. Makina mostly joins other SS2Ls to respond to ambitious invitations to tender a bid[24] Among Makina’s contacts we find Idealx, Nuxeo, Easter-eggs, Lolix, and others. These rather small-sized structures naturally join together to meet such large-scale projects.

To conclude this monograph, we asked the Makina Corpus consultants if they thought there was a bijection between "doing business with Free software" and "developing Free software", or, on the contrary,

[23] and, as a consequence, they are not afraid of job mobility and to use job portage facilities.

[24] We mentioned earlier that this could only be done with known individuals; here, the applicant can be known and recommended by another SS2Ls.

are there more and more parasites now.

Philippe: "For some, there is ambiguity in the way *free software* free software as opposed to software which is cost free. This can lead to a feeling of bijection, namely, people who believe they are doing business with something that is free [this is not correct], etc. As far as I am concerned, I do not see any ambiguity in the translation. *Free software* means logiciel libre, that is, a software program from which you can recuperate the sources. You can install it and be free to personalize it or not. Now, if it happens that you have neither the competence nor the time or desire to do so, and you decide to pay somebody to do this for you, I do not see anything wrong with this. Thus, there is nothing wrong either with the fact that people will offer to that for you. In fact, I compare this with working on old cars: I had some old motor bikes and I could have paid somebody to repair them so that they would work once again. This would not have shocked me, even if I had known that, just as for free software, the 'sources' of the mechanical assembling of the parts that make up this motor bike were freely accessible. This means that I would have paid him for something I could have done myself."

Mose: "The way this is formulated, I will say that one can develop free software but not make a living off of it. Making [money] with free software does not necessarily mean that that person must contribute to it, but rather exploit the capitalization of the community. There is definitely a notable imbalance here. However, this does not shock me more than other things that I observe every day. Anyway, this is what I think: if it is a matter of earning money with free software, one way to do that is to work with [Free Software] for a while, at least."

Part II

Economical Analysis of software production strategies of valorisation.

Free Software Service Companies: the Emergence of an Alternative Production System within the Software Industry?

MARIE CORIS,
University of Bordeaux Montesquieu, coris@montesquieu.u-bordeaux.fr.

If interest in the Free software movement has grown so extensively over the past few years, it is certainly due in large part to the diffusion of Linux. The commercial emergence of this executive system has permitted the creation of a market for all free software programs since they are no longer confined to highly specific niches and can be the subject of an overall implementation of an information system in the future. This interest and the subsequent commercial recognition of free software was initiated by Linux because it is an operating system. In other words, the operating system is the central element of any computer structures since it allows communication between software programs; between the software programs and the computer/machine and, finally, between man, the machine and the software programs (Dréan [1996]). However, the relative commercial success of Linux which, according to an IDC study, holds approximately 25 % of the server market shares[1], cannot be restricted to mere technological competition between software standards - although they are of utmost importance - since the production logic behind this impetus can be contradictory. Commercial or non-free software results from the exercise of intellectual property which allows those who control the market standard to have a monopoly over it. This situation can lead to lock-in (Shapiro and Varian [1999]). Nevertheless, the existence of free software is the consequence of an established scientific tradition of openness whereby the source code is made available to everyone. Thanks to the Internet, free software is the prerogative of the communities which develop software outside the commercial domain. Since Linux's operating system offered optimal technical reliability, making it possible to consider an alternative to Windows, distribution firms were created in order to market it. Many traditional actors in the computer field integrated it in their commercial offers. If the technical reliability of Linux and free software is often the fruit of the work of volunteer developers, the commercial recognition of the operating system can also be explained by the structuring of commercial, material and software offers which are permitted by the conjunction of the two following elements. First, it is necessary to take into account the creation of distribution firms such RedHat, Mandrake or SuSE whose sole vocation is to transform Linux into a commercial and user-friendly product with regard to its installation and use. Prior to this, Linux was only accessible to computer experts. Secondly, the existence of a free software market also owes much to the large number of traditional participants such IBM, Sun or HP who contributed to structure the offer of complementary products (Maume [2002]). This was accomplished through the portability of commercial software packages under Linux (Oracle) as well as through the portability of material components. A market niche can thus be created making Linux a true technological alternative to well tried commercial solutions such as those offered by Microsoft.

As a consequence, in addition to the creation of a market segment oriented towards "Linux", Service Companies Specialized in Free Software (SSLLs) were founded in order to offer companies the same services as traditional software firms such as Information Technology Service Companies (SSIIs)[2], except that they operate in the sphere of free software. The goal is to make free software development a marketing strategy, a new production system within the software industry, which will not be confined to voluntary communities of developers, but rather be based on the developments carried out by these communities. We are not talking about the possible replacement Windows by Linux or by some other operating system. This is about the emergence of a much broader alternative within the software industry. This alternative solution would be based on a new production system the purpose of which would be to try to supplant proprietary software development with free software. In other words, the question is whether or not the commercial success of Linux may be a sign indicating the emergence of an alternative system of production within the software industry itself.

Currently, the SSLLs are trying to gain a competitive edge within an industry which has been occupied historically by both SSII and software package editors. The SSLLs should not implement their production systems through the editors, but rather through the communities of developers themselves. In order to define this system and extend the business monographs that were suggested in the previous article and which were examples of the construction of such models, we submit a summary of the studies that have been conducted to

[1] http://www.idc.com

[2] In France, these SSLLs have been created since 1997 and are trying to compete with traditional Editors-SSIIs on the professional market. The acronym, SSLL, stands for *Sociétés Spécialisées dans les Logiciels Libres*. SSII stands for *Sociétés de Services en Ingéniérie Informatique.*

date. These studies are based on field work carried out in ten of the major French SSLLs between January 2001 and February 2002. These were accomplished in the form of semi-directive interviews with the top executives and the founders of these companies. The approach that was chosen in terms of the production system or productive model was developed by Boyer and Freyssenet [1995], [2000]. It is thus appropriate to define the production system according to the manner in which the SSLL has tried to handle the two fundamental uncertainties faced by companies: a) the instability of the market and b) uncertainties involving the work produced. Nothing guarantees the company that its capital investments will be profitable (i.e. that it will find outlets for its goods and services). By the same token, it cannot be absolutely certain that its employees will meet production goals. Research must be done at two levels when trying to understand how the SSLLs implement their production system: 1) identifying the competitive advantages that the SSLL can exploit (while taking into account the emerging competitive context) when faced with the historical domination of both the SSIIs and editors. This comes down to studying the target market segments and the services which are actually offered; 2) analyzing the organizational models which best characterize SSLLs. This can be summarized as follows : on one hand, the relationship between the communities of developers and the SSLLs and, on the other hand, human relations management within the SSLL.

6.1 Managing the uncertainty of the market: which competitive edge for the SSLL?

Whenever a business gets started or creates new products or services, it is subjected to the uncertainty of the market when it comes time to actually sell them. In order to counter this uncertainty, which is related to the productive environment in which such companies emerge, namely, the state of the software industry (1), the company must try to gain a sustainable competitive advantage vis-à-vis its competitors. To accomplish this objective, it must verify potential customers' credit worthiness with respect to their priorities* (Boyer and Freyssenet [2000]). This leads the SSLL to target specific market segments which seem advantageous within the framework of the competitive environment that has been identified. They must then offer a range of services based on a profit strategy which, in this case, relies heavily on highly personalized services (2).

6.1.1 How do the SSLLs fit in their competitive environment?

Linux, as well as the main free software companies, offer technological alternatives to the established proprietary standards. Although these software programs are innovations of specific products, one cannot talk about a technological schism here in the sense that new demands and new needs will neither structure themselves nor express themselves in the context of emerging market segments. In fact, other actors are already present where such software programs exist, like Microsoft which was dominating the market of operating systems for PC servers with Windows NT when Linux emerged. One also can think of the many applications which have been developed. Even if Oracle is carried under Linux, other free data-base management systems (PostgrepSQL, MySQL) are trying to position themselves to challenge the proprietary standard. However, Linux not only symbolizes free software but, as an operating system, it is the key element in the diffusion of all free software. Both the activity and especially the durability of the SSLLs are thus linked to the diffusion of Linux. The technological competition between Windows (the market standard) and Linux is of utmost importance in the study of the competitive environment in which the SSLLs find themselves. Analysis shows that it is characterized by both an apparently locked-in market and the creation of a market segment focused on Linux.

A locked-in market?

The very nature of software as an information tool allows a commercial software package editor to have a monopoly, at least for a while. (Shapiro and Varian [1999]). The exploitation of increasing returns of adoption (IRAs) can provoke the closing of the users' installed base[3]. the IRAs concern a central element, such as the operating system, all or part of the dependent industry can be locked-in as a consequence. Foray [1990] notes that the degree of lock-in will depend on the importance of the increasing returns of adoption. If one refers once again to Arthur [1989a]'s framework of analysis which was adopted by Foray [1989], [1990], it clearly shows that, in the case of operating systems, the five sources of increasing adoption returns exist:

1. Microsoft's dominance in the operating systems markets for micro-computers or PC servers can be partially explained by the economies of scale in production which allow for a temporary monopoly (hence the importance of the fixed costs and the weakness of marginal costs);

2. the economies of scale in production are reinforced by the economies of scale based on demand which are due to network outsources: the more a technology is adopted, the more it becomes useful to the user. This is a direct consequence of the increasing number of users' installed bases (sharing files, etc.);

3. in order to close the users' installed base and increase the network outsources, a generic software package editor can take advantage of systems adaptation costs which can be measured just like direct costs (i.e. new software acquisition) and spin-off costs (i.e. training). This training characterizes the confinement of a base that is installed on an operating system as well as the applicative software programs that is linked to it. Because the lambda user is used to a specific software package and because the editor guarantees a certain retro-compatibility between the successive versions of the software package in question, he/she does not see the point in changing the system because of the spinoff costs this would entail;

4. as for information goods, adaptation costs will rise proportionately in accordance to the degree of interdependence required by complementary products (i.e. applicative software programs, material components). This pertains to the concept of technological interrelationships. The higher the number of technologies that structure the environment of a specific technology, the more attractive the latter will be;

5. increasing information returns, that is to say, the more a specific technology is used, the more it becomes known, may explain why users are not prone to change. This is the case for private users and companies alike. Indeed, the more a specific technology is adopted, the lower the risks are and, consequently, the higher its diffusion will be. People side with the dominant solution for that very reason: because it is dominant. This is in fact IBM's advertisement: "nobody ever got fired for having bought IBM products".

In such a case, there is no longer a question of a temporary monopoly but of a real closure of the market. This not only concerns the users' installed base, but also an entire branch of industry which a company like Microsoft can control because of its monopoly of the operating system. To try to counter such a situation through the use of an operating system which would result from the same productive logic would only shift the problem to another proprietary solution.

In addition, and this could be a sixth point, intellectual property, which comes under the jurisdiction of national or supranational institutions, is of the utmost importance here. This is because it makes it possible to insert the SSLLs, like the software industry, within their macro-economic framework and to stress a

[3]The momentum by which a technology is diffused is self-reinforcing in the sense that the more it is used, the faster it will be adopted by a steadily growing number of people. This process of diffusion is formalized by Arthur [1989a] thanks to the concept of "Increasing Returns of Adoption".

more institutional source favoring lock-in. The GPL (GNU, General Public License), which protects free software programs, questions the legality of lock-in under French law (Clément-Fontaine [1999])[4]. In order for Linux and free software to develop, it is necessary for the public institutions to recognize them. This could be achieved through the simple adoption and use of these programs, since by using them, they gain legal recognition. However, right now, the most alarming question remains that of the possible patentability of software programs (Zimmermann [1995b], [1999]) and its impact on free software[5].

Thus, Linux is emerging on a market that seems locked-in by Windows. A finer analysis of this situation with regard to market segments (for example, private individuals with operating systems for client stations or micro-computers as well as the market segment involving professionals with operating systems for servers), shows that the server market is apparently not as locked-in as one might think. This means the situation is reversible, at least for informed users, and makes it possible to consider creating a "Linux" market segment.

Creating a market segment centered on Linux.

If one distinguishes the different types of users according to a simple criterion according to the technical control one has of the computer, one can distinguish two main classes: that of the non-specialists and that of the well-informed professionals. Most of the time, the lambda user has not benefited from any form of training - except that which is offered at school and which applies to the market standards. As a consequence, the lambda user seems to be more reluctant to take risks in comparison to a trained user for whom switching from one system to another will be much easier. In addition, one must remember that they do not utilize their computers at all in the same way. Most of the time, domestic users do not need to work on critical applications. They either want to enjoy themselves or use their computer to take care of their domestic finances. Therefore, they much prefer user-friendly software tools to highly reliable ones. This corresponds to what one calls a "mass market" which is particularly advantageous for commercial and user-friendly software package editors. The second category is that of the professionals, or at least well-informed users, who have specific needs and who thus can benefit from the type of free software Linux provides. For example, their professional training spares them from the spinoff costs deriving from having to utilize a new operating system.

In the same way that the degree of lock-in depends on the type of users concerned, it is necessary to pursue this analysis by distinguishing between two types of market segments for the operating systems: that of client stations, micro-computers and PC servers[6]. PCs were introduced on the market during the Seventies and were meant to be used for personal purposes. As a consequence, they were not designed to fulfill server functions. These were handled by Unix workstations at the time. Computer networking kept on increasing (i.e. businesses' internal and external networks), and it became necessary to realize the needs in the number of computers owned by companies. At the beginning of the Nineties, PCs were technically modified in order to become servers (thus, the term PC-server was coined). Since Microsoft already had a quasi-monopoly on the operating systems market for micro-computers, Windows' NT version for servers became dominant in this new market segment. However, two elements relating to the emergence of Linux should be taken into account which relativize the extent of the monopoly and the possibility of a closure strategy in this market segment. On the one hand, PC server market segments constituted a new market at the time when Linux came to the attention of the decision-makers of the computer world. This means that, if there really is a quasi-monopoly, the situation can still be reversed. On the other hand, if PC-servers quickly spread and become more affordable, because they quickly attain the performance-level of workstations, businesses will want the operating systems for these PC-servers to have the same technical characteristics as Unix. Linux is nothing else but "Unix for PCs". Taken jointly, these two arguments can explain why the diffusion of Linux as an operating system for

[4]The GPL excludes any responsibility on behalf of the software program authors. Distribution companies overcome this problem by introducing a guaranty clause in the licence contract.

[5]According to certain authors, the patent could be detrimental to the future of free software and consequently that of the SSLLs (Cf. Faucon and Smets-Solanes [1999]).

[6]This analysis owes much to the work of Jullien [1999] in this domain.

PC-servers was carried out more rapidly than in the client station market.

However, the fact that lock-in seems to be less irreversible in the professional market, and, particularly in the servers market segment, is not enough to explain the successful diffusion of Linux. Indeed, what would be the point for a user to choose a solution that was strictly equivalent? It is necessary for a new technology to offer something that encourages the user to adopt it. As far as Linux is concerned, it offered a highly reliable operating system (Horn [2000b]) and this was a determining element in its adoption. Because free software programs are famous for their technical qualities and reliability when it comes to implementing critical applications (Apache or Linux undoubtedly demonstrate such assets), they meet the needs of professional users in certain fields where reliability seems to be a determining element (embedded systems, real time, etc.)

This analysis which was conducted at a theoretical level, has made it possible to understand the context in which SSLLs have emerged since 1997. Experience has confirmed our analysis. However, to analyze only two products which are competing technologically would not have been satisfactory. The diffusion of Linux could not have occurred without the contribution of distribution firms or the support of some best-known computer companies. All of the SSLL managers that were interviewed agreed on the fact that, at the very beginning, free software programs were lacking a sort of professional aura for potential customers. In addition, since the operating system is the central element in the information system's architecture, technological interrelationships are of prime importance and can allow a way out of lock-in. By ensuring that hardware and software is compatible, companies such as Sun, HP, IBM, Oracle[7] significantly reduce the direct adaptation costs as well as the RCA source relative to technological interrelationships. The compatibility of applicative software which allows files or data to be exchanged regardless of whether the software is used under Windows or Linux, enables the effect associated with network outsources to be less significant. When the first SSLLs were created, compatible software packages hardly existed. They have been becoming more and more numerous ever since. In the beginning, such markets were confined to a very restricted number of applications (Web servers, for example). Today, an increasing number of applicative software programs are available and reduce the adaptation costs in terms of fields of application. However, supplies remain quite poor in domains such as accounting. When a company like IBM joins in, it provides Linux and other SSLLs with the credibility they need. The executives do acknowledge that IBM wants to get a share of the market and push its hardware offer through the investments it has made on behalf of Linux[8], through its integration in the offer of the firm (Lin [2002a]) and through an aggressive publicity campaign in favor of Linux (i.e. advertisements broadcast on Hertzian channels).

This is how a market segment is created around the Linux operating system. Because their activity depends on the diffusion of the operating system, SSLLs will try to fill the slot and compete with SSIIs (Information Technology Service Companies). Because a professional market is more difficult to close, the diffusion of Linux will be faster and, consequently, SSLLs can expect to make real profits.

6.1.2 How do SSLLs fi t in? Competitive advantages, market segments, types of services.

With the creation of a "Linux" market segment, the operating system attracts the attention of companies which, henceforth, consider it like any other product. This does not mean that we are dealing with a new production system but it does show that there is another technological alternative. Until now, Windows was the standard. Distributors can play the role of editors for Linux and SSIIs. One of their functions consists of integrating software packages provided by editors and adapting them to the specific needs of the users. Because their activity merely revolves around the free software industry, SSLLs are paving the way for an alternative system of production within the industry. We are going to study this system by examining the kinds of competitive advantages it may offer to the SSLLs and their "product policy", that is to say, their trade and the services they

[7] It is necessary to relativize this matter. Software program architectures are very complex and many are so-called "job applied" software programs which are not carried under Linux or which do not have any equivalent in the free software domain.

[8] On this subject see IBM's site: http://www-1.ibm.com/linux/.

actually offer.

Competitive advantages thanks to low costs and reliability.

SSLLs are emerging on a market which targets customers who depend on other service providers: the editors, who supply both applicative software packages and software systems, and the SSIIs which meet the customers' requirements as far as integration, adaptation, software personalization or specific software applications are concerned. Since they arrive on a market which has been historically dominated by the Editor-SSII pair, SSLLs must offer competitive advantages. What kind of strategies can they exploit to attract customers and what kind of market segments can they fill?

In the same vein as Linux' distributors, SSLLs will play on their reliability which is intrinsic to free software programs. If Linux is so popular, it is not because it embodies certain IT ethics, but because it has extremely favorable echoes concerning its technical reliability. However, we have seen how lock-in can exist when a technology that customers may favor is not selected because another technology has previously been highly diffused and, as a consequence, adopted by nearly everyone. Such an attitude leads to an extremely passive market, where everybody waits for his neighbors to adopt a new technology before doing so himself[9]. For software programs to be adopted, they need to be highly diffused: customers must be targeted very thoroughly. The SSLL executives who were interviewed and whose companies had been on the market since 1997-1998 admitted that, initially, they had aimed at the industrial S&Ms market. They finally decided to concentrate on industrial heavy-weights and government agencies. In this case, field work seems to validate the theoretical assumption that it is vital to offer technical quality in order to attract those for whom reliability is a critical element. Once this is done, other customers will follow suit on their own[10]. This is why SSLLs have relied on Linux's technical advantages in order to market this program to companies which demand optimal reliability (embedded systems, real time applications, etc) (Maume [2002]). RCA sources must be reduced as far as reluctant users are concerned. It is essential to find out the category of users who are risk-takers. Thus, the fact that the industrial heavy-weights are the first to adopt such programs is not surprising. They have extremely competent personnel at their disposal and this facilitates the migration of their servers under Linux[11]. A second factor has to be taken into consideration: that of "backward induction", namely, the smallest companies which are aware of technical advantages of this phenomena tend to follow suit and adopt such programs too.

Yet, purchasers are not only sensitive to the technical reliability of a solution, but also its price. This is a key factor to whether or not they adopt Linux programs. Many companies have adopted Linux because of its low license cost. Although the majority of SSLLs adhere to free software programs from an ethical point of view and see a major advantage in making their source code available, they use competitive prices in order to favor the diffusion of free software programs. From a financial perspective, the competitive advantages from which SSLLs benefit, for example, by re-using software programs which are license-free (integration, adaptation...), have repercussions on the services customers must pay for. Indeed, this allows potential clients to adapt their systems at a reduced cost. Such savings seem to facilitate the shift to Linux and favor the adoption of free software programs in general.

The SSLL executives who were asked about customer reactions with respect to free software in general and LPG licenses in particular, confirmed that large companies are the least nervous about adopting free software insofar as achieving high technical levels is what matters most to them. When shown advantages in terms of license costs and reliability, these customers remain more or less indifferent to the type of license that has been selected. Today, Linux seems to want to attract S&Ms, but this type of market is only emerging now. If SSLLs focus on important markets and government agencies, it is clear that they are not going to

[9] This is what Farrell and Saloner [1988] call "backward induction".

[10] For a technological model of diffusion of Linux, see Dalle and Jullien [2000] and Jullien [2001]

[11] Recall the Information Technology premise whereby only important companies and the government administration were computerized.

be interested in micro-computers where lock-in is a lot more commonplace. Indeed, although discussions are underway, the relevance of targeting client stations, Mandrake's mistake was precisely to bet on Linux's user-friendliness and easy installation on client stations, although this kind of market did not really exist. To simply sell packages did not prove profitable: companies need a whole range of computer services (training, maintenance, integration, development), whereas the lambda user wants products that can be used as just as they are. If selling utilization licenses on a mass market seems viable enough, private clients do not fit in SSLL markets even though they might not be SSII clients either. Thus, powerful companies remain the most adequate market, with a server-oriented approach.

If the SSLLs' competitive advantages in terms of reliability and costs allow them to focus on the right market segments, they must still choose the type of services they want to offer.

The SSLLs' product-policy: identifying the most profitable services.

The first thing to avoid is a break in technological continuity since the objective is to have customers switch to equivalent new solutions in terms of functionalities. In addition, competitive advantages must be offered in terms of reliability and costs. What kind of solutions can be monetarily viable for service companies which specialize in free software programs? These can only be marginal solutions since the Editor-SSII's traditional models are number one on the market.

The French SSII and software program Editors' Union, Syntec, has drawn up a nomenclature of the services they offer in order to define and identify the French SSIIs. It has listed three service categories: 1) "intellectual" types of services (counseling, auditing, studying computer systems architecture, integrating new systems, engineering, maintenance, technical assistance and training); 2) services linked to machines (networks, value-added services and information management); 3) services linked to the design of software packages (Mouline [1996]). If we consider this type of identification, it is clear that SSLLs fit in the same category as SSIIs because they offer services which are basically identical. However, SSLLs seem to want to dissociate themselves from SSIIs. As a consequence, Alexandre Zapolsky, CEO of Linagora, has chosen to change the term from SSII to SSLL (i.e. *Société de Services en logiciels libre* meaning Free Software Service Company)[12]. This has allowed him to maintain the same acronym for the two branches (since both offer services) and yet, show their differences by stressing a specific field of expertise, in this particular case, free software programs. Although this is not a unifying term, the SSLL executives insist on the need to fill the slot for free software programs and are proud of doing so. If such a step had not been taken, the competition that already exists between these two types of company would have gotten even worse. The SSLLs thus hope to attract companies which are interested in a shift to Linux. By highlighting their expertise, they hope to be selected when bids are made.

Indeed, analyzing the different interviews of SSLL executives mentioned above, clearly shows that the their main competitors are indeed the SSIIs, not the SSLLs. Although SSLLs do compete with each other, this competition seems to be constructive, at least for the time being, because it sends positive signals to the market. We must keep in mind that the SSLLs' aim is to gain a professional aura. Potential clients tend to be reassured when there are a number of different actors on the market. Therefore, the SSLLs benefit from these positive outsources which are generated by the mere presence of their competitors. By the same token, and according to what can be gathered from these interviews, the SSLLs' *raison d'être* is not "to make money with free software programs". They want to develop a new production model within the software industry. For example, when Richard Stepniewski, director of Adelux, was questioned on the degree of competition that exists between the various SSLLs, he used the word "coopetition" meaning that, if there is competition, the companies often cooperate with one another in order to bring credibility to the free software industry.

[12] SSLL is trademark registred by Linagora. Since, the term has became the usual term to name such companies in Information Technology journals (*Le Monde Informatique* of January 25 2002) and is increasingly used by service companies surveyed themselves. That is the reason why we have chosen to keep it to name them.

Unlike SSIIs, SSLLs do not offer editing activities in the strict sense. SSIIs devote part of their activities to the adaptation of existing commercial software packages for the benefit of their clients (this is done through licensing agreements made with the editor). However, they also develop entire solutions which can be more or less specific. Because SSIIs often focus on specific lines of business, they develop, market and adapt distinct job solutions (for example, specific software packages for the management of territorial authorities). To some extent, this is how they make money: they sell user licenses and services that are linked to the adaptation, maintenance and evolution of the software. SSLLs cannot make money by selling licenses since editing activities do not exist in the free software industry. They cannot charge users for something they are supposed to get for free. At one point, the Aurora company did try to develop a special "deal" on its products by selling licenses. It became a GPL software editor. However, the experiment proved to be a failure: free software licenses were not compatible with the editor model. The company began to cumulate financial losses due to the lack of profitability of its editing activities and fell back on its primary function, that is, services. The company got back on its feet and was bought by Systran in August 2001. Jean-Pierre Laisné, founder of Linbox company, went through the same ordeal and went out of business in June 2001. He is now the director of Open Source strategy at Bull and states that, following his failure (he also wanted to sell Linbox products in the form of licenses), the only viable strategy, profit-wise, is to sell services. The recent refocusing of distributive firms such as RedHat, Mandrake, on service activities (Lin [2002b], Lin [2002c]) clearly show that it is not monetarily advantageous to confine oneself to selling only Linux packaged versions.

In order to compete with the major actors in the field, SSLLs maintain that it is necessary to rely on the extreme personalization of the services by exploiting development economies of scale because of the free re-use of software components. This is the big advantage Linux and other free modules have: they precisely allow such huge flexibility thanks to the provision of their source code. The recent evolutions in the software industry have lead Horn [2000b] to identify two conclusions which are accepted by the professionals: 1) the needs of the commercial software packages companies are not satisfied (many functionalities are useless, thus unexploited); 2) users need customized services more and more. Modularity enables this kind of customized service, and it fits in beautifully with the logic of the free software industry since its components can be modified. As the software industry continues to head in the direction of services (Eurostaf [2000]), SSLLs are making the most of this context. Excellence can only derive from quality services which also offer a great diversity of programs which meet the very specific needs of the customers (Horn [1999]). SSLLs count on the perfect adaptability and interoperability of their free components to diversify their offers according to the requests they get. The assets of the free software programs - reliability, flexibility, portability and compatibility - make it possible to have excellent modules which, in turn, allow them to find the best solutions for their customers (Horn [2000b]). The interesting point in the development of free software programs precisely consists in having a common base that can be adapted to each and every request. When a company uses free modules which have been developed by another one, it does not only save time but also money since the module has already been designed. The company can thus make profits faster and personalize its offers as it sees fit. Excellence comes by selling expertise and time, not licenses.

This is no longer a matter of selling on a mass-scale, which is what software editors do by taking advantage of the economies of scale, but rather of benefiting from the economies of variety in order to offer products which are different, yet based on common software components. The value of the product gets higher thanks to customized services and software developments which, in turn become profitable for the SSLLs. It is primordial to create long-term contracts of confidence based on the development of customer loyalty which, in turn, stems from interpersonal relationships.

All the SSLLs establish their profit strategy on diversity and quality. On its own, Linux would only be another operating system, but the logic of the free software industry is to set up an alternative production system within the software industry. We have seen how the uncertainty of the market can be apprehended.

One question remains, though: the production of the goods and services that are offered. There are no guarantees that the company will know how to implement the production of its services or manage to have its

employees meet deadlines.

6.2 The implementation of the production system: the sslls' Organisational Specifi cities.

We have seen how SSLLs can fit in some market segments according to the type of competitive environment they find themselves. In the same way, we have identified the types of services which seem most profitable for them. However, for a company to identify a satisfactory profit strategy and product-policy is not enough. It needs to determine its system of production. If the company's executives view their system of production as a "model", they must choose coherent means of implementation for their product-policy. This brings them back to their productive organization, that is to say, the means and methods that have been chosen to conduct their service production (1) as well as the relationship they want to establish with their employees. A compromise must be found between them and their executives in order for the employees to meet production requirements (2). These two innovative elements, organization-wise, are distinctive features of the SSLL production systems. Our analysis is strictly empirical.

6.2.1 Organizing the production of services - SSLLs: between the community and the market.

Many authors have analyzed the way communities organize themselves (Lakhani and von Hippel [2000], Lerner and Tirole [2002], Jullien [2001][13]) and one might be tempted to imagine or wonder how such a model can be transposed to companies. Even if it is true that a company can model its organization on that of free software projects, the idea of transposing the organizational model of a communities is simply unrealistic. Field work has shown that the key element of success would be to respect the communities, because the SSLLs' expertise leads them to function as an interface between communities and clients. If the "give and take" model makes it possible to better understand the way things work within the communities (Lakhani and von Hippel [2000]), the SSLLs could not subscribe to it in the implementation of their production.

The SSLLs' expertise: an interface between communities and customers.

At present, the free software industry is filled with software-project programs that are developed by entire communities of often voluntary computer specialists. Few free software programs are the fruit of strictly commercial projects (Zope, GNAT, OpenCascade..). As a consequence, it is not always possible to have an editor who could be associated with the software components that are used and re-used by the SSLLs when they implement their customers' projects. These companies must then serve as an interface between the communities of developers and the customers, just as SSIIs must serve as an interface between the editors and the customers. The role of the SSLLs is to guarantee the level of expertise of the free software programs on behalf of the customers. The degree to which this expertise is as a binding element between the communities, the SSLLs and the customers is expressed by Christophe Le Bars, Alcove's founder and former technical director. In his eyes, the role of his company, as well as that of any SSLL, consists in being "a mediator between the community and the companies", insofar as it also entails providing "editor services" for free software programs. He reminds people that editing does not simply consist in writing and marketing software programs, but that 50 % of their activity also consists in providing expertise with regard to their software packages. Alcove's goal is to position itself within this " 50 % ". Indeed, since these software programs are mainly the work of communities they cannot be tied to any publishing house which could guaranty their expertise. Within the software industry, there is a very strong bond with the SSIIs which play the role of integrators. SSIIs focus on their clients' jobs and meet their needs at a personal level by utilizing software bricks which are the propriety of the

[13] Many references are available on the following site: http://opensource.mit.edu/onlinepapers/.

editors. Editors who provide part of these software programs generally guaranty their technological expertise. This does not mean that SSIIs do not propose pure technical expertise on modules other than their own, but rather that this is not their primary role. On the other hand, this should be the SSLLs' responsibility since, following this model, it is the communities who are the editors and these communities are not recognized as entities that can provide such expertise. SSLLs serve as a bond between the communities which develop the software program source codes (i.e. raw code) and the users who make use of this code in its executable form. Since anybody can use these free software programs, companies could help themselves as they see fit (via Web sites), without having to pay for services. On the one hand, companies do have adequately trained computer specialists but they often lack time to analyze and expand such solutions since this is not their primary job function. On the other hand, free modules are available in their raw form. License agreements are a sort of convention which structure the organization of the development within the communities but which generally do not mention any warranty clause, not even with regard to the proper functioning of their products. It then seems too risky for any company to build their computer systems on solutions which have not been *previously* tested. This is where SSLLs come in to improve these software programs with their own expertise and, thus, make them marketable. This includes warranty clauses, training, maintenance, etc. In short, they make these software programs "credible".

Other than providing expertise, SSLLs provide the same services as SSIIs, namely the integration of software solutions within their clients' information systems. They also offer a customized software programs development. Since modules are open to anybody and can be freely modified for SSLLs, these companies organize and base their production on components that communities put at their disposal (Jullien [2001]). SSLLs can solve the paradoxical type of modularity which is being used within the software industry by making the most of the fantastic flexibility that free licenses allow, flexibility which derives from the modules interoperability. On that subject, Horn [2000b], like all the executives who were interviewed, remember how much people were confident about components libraries in the 1990s. Using pre-existing software bricks was supposed to save time and money since, in order to rewrite entire software source codes, it was not necessary to start from scratch anymore. However, following this logic, the bricks or components remain under proprietary licenses, and the APIs (communication interfaces between various software bricks) remain the property of the editors. When it comes to free software programs, everything becomes accessible and one can work freely on a component according to the customer's requirements. With such fantastic flexibility, it is possible to offer customized services and, consequently, to carry out the product-policy that we have previously identified.

Software programs are information goods and their extremely high design costs are a barrier that makes it difficult to challenge the monopolies. The economies of scale which result from this situation (due to the quasi-inexistent marginal replication costs) seem to be the first source of increasing adoption returns which favor lock-in. By re-using the work of the communities, SSLLs are no longer subjected to design costs since the development of software programs is based on what already exists[14]. The first source of IRAs is thus neutralized and free software programs can offer a technological alternative to the market standards, because they result from another development strategy. Low license costs combined with low design costs constitute a big advantage for SSLLs when bids are submitted[15]. However, such voluntary work completed outside the commercial sphere could not be realized without something being given in return.

Collaboration with Communities of developers: advantages and limitations.

If SSIIs must pay royalties and other access fees to use software package bricks in an effort to respect the legal constraints of "editing" regulations, SSLLs compensate by developing free software programs. The aim of the community of developers is not to diffuse the software but to write it (Cohendet et al. [2001], [2002]) in accordance with a software development method which favors the quality of the source code rather than the

[14]This is how Linux "distributors" went about it.

[15]The invoicing of services is generally based on the time spent.

more expected goal of profit margins normally associated with commercial editing (Mateos Garcia [2001]). Even if the commercial diffusion of free software brings recognition to the developers (Lakhani and von Hippel [2000]), this is not their main objective. Above all, their primary objective is to have everyone accept the idea that opening the source code is the same as permitting the open exchange of ideas, an ethical stance that is shared by the scientific community (Himanen [2001]). In exchange, they offer to contribute to community projects, whether this assistance is technical in nature, involving the participation in the development of software, or contributions in the broader sense of the term, namely, making free software the dominant mode of software development.

In this way, and through respect for the GPL, improvements made by the SSLLs will show up in the common effort made in producing free software. Beyond showing respect for the terms outlined in this specific license, the SSLLs can contribute to technical developments by participating in diverse projects. This can be accomplished by committing a some of their employees' activities in cases where, for example, they can specialize their offer in the field of free software. Thus, they clearly indicate that they are greatly interested in participating in the development of a specific software project in order to make it efficient and, consequently, more appealing in the eyes of the client. They are also interested in having the community which is associated with the project test the improvements or corrections that have been made. Anybody can check the type of work the SSLLs do on certain free components on Web sites linked to the project. However, they do not name these developments (software components, patchs, etc.) after the companies themselves because of their will to show openness with respect to the communities. Since the SSLLs want these software programs to become the standards on the market, they avoid attaching the name of the company to these programs so that they are recognized as the fruit of collective work and so that the communities will continue to contribute to this work. Some companies, such as Alcove and Easter-Eggs, have Web-sites - they often have an .org site which differs from their .com site which is a strictly commercial site - that are primarily dedicated to free software activities where one can find information (i.e. articles, definitions), but also a kind of virtual "development laboratory", with development projects which are specific to SSLLs in the form of free licenses[16]. There are two reasons for this: 1) the companies want the communities to realize how much they contribute to the different projects; 2) the companies also want their customers to know that they are truly involved in the free software universe. Let us keep in mind that SSLLs are not attached to any publishing house, thus, do not get any expertise or training. This is something they must acquire on their own. Being part of such community projects seems to be the ideal way of maintaining their level of expertise.

Following this logic, SSLL executives want to make it clear that they do not regard the software as goods but rather as information (which is why software patentability is refused). What needs to be protected, in fact, is not the source code but the companies' own data, which can be guarded by cryptography. The SSLLs share the same culture and ethical principles as communities do. Actually, both their executives and their employees have often worked in those communities before. Isabel Zwegers, Asynux's chairperson, specifies that there are two ways of contributing to the community. The first consists of the "technical" collaboration in free development programs; the second consists in working in accordance with the free software ethical stance. "*By creating a company on such a model and by trying to make it credible market-wise, by demonstrating it can be profitable, it then becomes possible to prove that the communities' ethical stance is viable within the commercial world.*" There can often be fluctuation between these two approaches: companies do not take part directly in the projects of the communities but rather transform the company projects into General Public Licenses (Savannah with Alcôve, Midgrad with Aurora, Camtrace with Axis and Agix, etc.). Others share Zwegers' point of view. For example, Jean-Noel de Galzain thinks that "*if, in theory, free software makes it possible to improve profitability (productivity and competitiveness) and service quality, companies must prove that such a model exists and that it is viable.*" Linagora's CEO, Alexandre Zapolsky, confirms that the most important challenge for the SSLLs and free software in general is to be successful at an economic and social

[16] Philippe Breider, sales manager for the SSLL named Easter-Eggs, states that these Web sites are generally not intended to create a community of developers who are associated with project outside of the company.

level. Right now, they are technologically successful.

However, the tacit partnership that exists between the communities and the SSLLs shows some limits. The SSLLs seek to support the diffusion of their free license developments, but they must meet their clients' demands, clients who are still reticent. As a consequence, the SSLLs frequently ask for proprietary licenses. Their primary objective is to have customers adopt free software programs such as Linux, Apache, and so on by showing them some of the benefits which have already been identified. Then, they must convince the customers to accept that the developments carried out for his/her sake are protected by GPL-type free licenses. However, because free licenses are wrongly mistaken for "free of right" licenses, the customer thinks that the best means of protecting the company's data is to close the software source code. He/she thus has a tendency to favor proprietary licenses. Therefore, the SSLLs, with the support of both the communities and the associations, must use lobbying strategies vis-à-vis the general public and the national or supranational institutions. Because the SSLLs are at the customer's service, the expertise as well as the other services which are offered, such as customized developments, cannot be limited to the sphere of the free software industry. SSLLs do propose proprietary licenses since the client is the one who decides in the end[17]. The type of relationship the SSLLs develop with their customers can then be badly perceived by the communities. The latter, which are sometimes seen as the free software industry "fundamentalists", get the feeling that all the SSLLs want is to do is to get control of their software code. This could hinder their mutual collaboration. Moreover, Mateos Garcia [2001] clarifies the differences which exists between companies whose main objective is to make profits and the communities whose goal is to achieve technical quality. The latter regard software writing as an art. These two groups definitely show divergent philosophies. The SSLLs must meet the customers' requirements, while the communities respond their own needs. As a consequence, one notes that the jobs that are available in the field of free software are not adaptable. Thus, we come back to the question of whether or not economies of scale lead to lock-in which can, in turn, cause a sort of hybridization of computer systems that include either free or proprietary elements. At this stage, the "the free software industry economy" still seems too recent and too dependent on the work of the communities. As for commercial participants joining in, they still seem to focus too much on Linux.

6.2.2 Manage the uncertainty of work: an arbitration between rules and freedoms.

We have seen the type of job the SSLLs can do and to what extent their collaboration with the communities seems paramount in the production of services. It remains to be seen how the SSLLs can implement this type of production from within, namely analyzing their relationship with their employees. Nothing guarantees the SSLLs that their employees will meet the production objectives in due time (Boyer and Freyssenet [2000]). It is thus necessary to examine how the communities and employees can co-operate. In doing this, the special nature of free software employees must be taken into account. There has to be some sort of arbitration concerning rules and freedoms. We will illustrate this compromise by studying the Alcôve example.

The characteristics of free software employees.

Our objective is not to transpose the organizational model that is specific to the communities of developers. We will inspire ourselves from it, however, because we must take into account the bond that exists between these communities and the SSLLs through the necessary collaboration that we have previously analyzed. We have also seen that SSLL employees have often worked in these communities in the past and, as a consequence, share the same ideas with regard to source code sharing. The different interviews we have conducted demonstrate that there is nothing revolutionary in the way SSLL employees work. The principal difference between

[17] In this regard, Alexandre Zapolsky, Linagora's CEO, recalls that the principal advantage of the proposed free software alternative is to permit the client to choose between equivalent solutions - free or proprietary - rather than being strictly limited to only proprietary licenses.

the traditional model (SSIIs) and that of the free software industry lies in the presence of the community which generates another way of organizing work.

Indeed, the employee management model that is used in the SSIIs cannot be the same as the one used in the SSLLs since the values of the latter are closer to those shared by hackers (i.e. the code must be revealed). Relationships are based on the freeing of both ideas and source codes. There simply could not be any retention of information. Richard Stepniewski, Adelux's CEO, explains the reasons that pushed him to transform Adelux into a legally independent company. Adelux used to be part of Adequat, an SSII company. He claims that the management of human resources is different and adds "*Linux personnel are not handled the same way as in traditional SSII firms*". In SSIIs, work is partially based on the retention of information. Employees do not spontaneously diffuse what they know since that knowledge is what makers their work intrinsically valuable. It also gives them an advantage over other employees. Richard Stepniewski agrees that it is difficult to have these two types of technicians (SSLLs' and SSIIs') within the same structure. Jean-Noel de Galzain, Aurora's CEO, notices that, in the SSLLs, one finds the community model which is based on the following two concepts: "share and collaborate". Just as there are no editors in the free software industry, there are no training courses organized for SSLL employees, contrary to SSIIs. Once again, the principle which favors the free flow of information dominates insofar as every employee must share the knowledge that has been gained with his/her colleagues. This can lead to internal, more formal types of training courses.

Contributing to communities' free software projects makes it possible to partially guarantee the employees' training since it is then considered to be "self-training". Because employees often come from these communities, they keep in touch with them. This makes collaboration much easier. "Self-training" will be partly done when the employees are between contracts. It is also during these same periods that they are on the look-out for new technologies. This element seems common to all SSLLs even if some of them have a specific R&D pole.

The characteristics of SSLL employees result from their "hacker" ethical stance. According to numerous SSLL CEOs, this allows them to hire and pay their new employees lower wages than is usually the case. The primary motivation of these employees is to work for the free software industry, not to find the best job in terms of salary. Yet, this characteristic presents a certain number of disadvantages. Employees may think that to work in the sphere of free software is the same as working within communities.

Within the communities, those who volunteer to do a specific job work on the parts of the code which they control or enjoy best without any time constraints. If projects such as Linux become so important, it is because thousands of developers focus on them. The stakes at hand are high and exciting enough for everyone to try their best, so that, in return, their peers acknowledge the work they have done. SSLLs have less human resources and employees remain at the service of the customer. Here we see the principal difference between communities, whose aim is to produce state of the art technology, and companies whose primary aim is profit. Indeed, if the SSLLs want to show that the free software industry is viable as an economic model, they must prove this by showing profits. For example, in order to meet time constraints in the implementation of projects and favor profitable activities for the company, it is necessary to impose a certain degree of hierarchical order.

The key idea that keeps coming back in all the interviews is to make the employees "responsible". This is the only approach companies have to handle work-related uncertainties, contrary to the division of work intelligence, a historically dominant approach (Boyer and Freyssenet [2000]). This could not be accomplished without imposing a certain amount of hierarchy, the aim of which is to structure the company as well as to help guide employees. Sometimes, the free flow of information must be limited to the company. This can be jeopardized by the fact that various SSLL employees meet outside of work and share the information they hold, according to their respective cultural traditions. But it is also possible to make employees more responsible thanks to flexible work organization. This can be achieved in a different manner according to the size and ethical stance of each SSLL and must be based on a contract of confidence with the employees. It is necessary to find the middle road between rules and freedoms that can be accepted by both parties: the executives and the employees. Letting employees take part in their personal projects makes it possible to manage inter-

contractual periods and to maintain a high level of expertise which, in turn, contributes to optimizing the company's performances. However, such freedoms are neither introduced uniformly nor universally. There is a sort of "give and take" that occurs between the time that is allotted to each employee and the time that this same employee "owes" to the company. If participating in the communities' projects allows the employees to have some form of "compensation" for the developments which they have realized, this compensation should be given only if it brings something to the company in return. It is clear that the employees must be the ones to decide what is a leisure activity and what is work.

Coming to a compromise between rules and freedoms: the Alcôve example.

This SSLL has tried to establish a compromise concerning rules and freedoms based on its own experiences and a "trial and error" approach adopted by the company since its creation in 1997. At the beginning, for example, there were no schedules. People worked whenever and wherever they wanted: at night and/or at home. Since their work is purely intellectual, they can spend long periods of time thinking or doing research without anything being produced per say. It is difficult to judge the employees on the work they are actually carrying out. Executives want their employees to work wherever they feel most comfortable so as to increase the productivity of the company. As the SSLL expanded, and consequently the number of its employees, the customer demands had to be met (a customer assistance service had to be provided during " business hours") and, as a consequence, schedules had to be introduced. The goal was also "to socialize" work, that is, to force the employees to meet so that they would not "close themselves off" from others as can be the case for some employees who do voluntary work within communities. However, Christophe Le Bars from Alcove insists on the fact that he still does want to exert any control over his employees. He does not check what they are doing or when they are doing it. The employees manage their schedule as they see fit. The employees are judged once they have completely finished their work. The results are what count. They are given a certain amount of time to carry out their mission at the end of which they are evaluated. What matters then is the information flow that helps them communicate and solve problems. Whether employees contribute to their own projects during their working time is neither controlled nor sanctioned because they often work on business projects during their leisure time. There is a fine line between what concerns the employee's work and their leisure time. It works just the same between constraints and freedoms. All the SSLLs seem to agree on the subject of work flexibility, whether this has to do with schedules or organizing personal work.

As far as Alcove is concerned, the employees devote between 10 % and 20 % of their time to independent projects[18]. Nevertheless, Christophe Le Bars stresses that the 10 % to 20 % mentioned, concerns the total company time, not that of each of its employees. This time is not distributed uniformly, but according to the trust an executive can have in his/her employees, according to contractual arrangements and, finally, according to each employee's aspirations. This approach can also be seen as a reward to compensate employees for their relatively low wages (as is the case for Easter-Eggs' employees). The employees work on something they really enjoy and are still paid for it. However, the employees' participation in the company as well as their incentives may differ. When they recruit marketing people, SSLLs seek those who are experienced in this field, not those who have plenty of technical training - which is the case of computer specialists. For the latter, there are monetary compensations. On the other hand, one of Alcôve's employees, who is also the president of APRIL (Association for the Promotion of Free Software), has a flexible schedule which enables him to officially include his association's activities within his schedule. This makes it possible for Alcôve to enhance its image in the eyes of the association, but also to benefit from its employee's competence. He maintains that he could ask for much higher wages in a traditional SSII, but this is not his primary goal[19]. As far as computer specialists are concerned, although they represent the highest number of employees within the SSLLs, they

[18] R. [2001].

[19] In this regard, the example the SSLL known as Easter-Eggs can shed light on the matter, although it is a very particular case, where all employees own the company (via an association) and have the same salary, which was only 10,000 Francs net in August 2000 (€1524), and FRF12,000 (€1829) in December of the same year. See http://www.esater-egss.org.

are not necessarily experts in free software programs. Jean-Noel de Galzain has noticed that even if they eventually get such expertise, he tries to recruit qualified people who are open-minded and capable of working on proprietary types of technologies. It is not unusual to find quite a few former Unix experts (Axis and Agix's) in the SSLLs. They actually choose to do so on their own and switch towards Linux and other free software such as Asynux because of the quality and openness of their system. This remains within the tradition of Unix. The compromise between the rules and freedoms that can be found within the various SSLLs is based on the idea that *"one does not grant freedoms simply to create fantastic working conditions, but rather to obtain a higher productivity. It is easier to reach such an objective when a certain amount of freedom is given"*. Most of the CEOs interviewed agree with this statement [20].

If the SSLLs are aware that profits can be gained with the free software industry, it is not their objective to take advantage of the communities without giving something in return. In the same vein, if they believe in this free software industry as a sort of ideal model, they must prove its viability in a pragmatic way, by showing both profitability and serious work.

Conclusion.

The broad outlines of the production system implemented by the French SSLLs has been presented. Although this system seems viable enough for the time being (after one year of existence), the future of the SSLLs seems quite uncertain[21]. It would be premature to envisage a worldwide extrapolation of this system of production within the software industry[22]. On account of the fact that they are the entrepreneurial fruit of the communities' ethical stance, the SSLLs constitute a group of very particular economic actors. Indeed, they are trying to give rise to a system of production which is exclusively implemented in the sphere of the free software industry. This minority movement has emerged in a context which seems to favor this approach as opposed to that of proprietary software. There seems to be a will to extend the protection of the intellectual property (for example, acknowledging the patentability of the software at a European-wide level). That is why, by joining associations which promote and defend the free software industry, and by participating in various demonstrations, which sometimes have a militant touch (Free Software International Meetings, etc.), the SSLLs take part in the lobbying movement which incites governments to recognize and use free software programs within their public organizations. Associations such as the SSLLs then try to reduce the uncertainty of the market by acting directly on their institutional environment in order to influence the decisions of the participants at a micro-economic level.

If the free software industry seems to be making headway in this domain[23], it still appears to depend too heavily on the work of the communities. Beyond the cohabitation that would occur between free and proprietary software elements, such dependence could lead to the hybridization of both the products and the systems of production. Moreover, we have seen before that there is a great void in the line of business linked to free software[24]. This generates a hybridization of the companies' information systems when the latter are composed of both free and proprietary software programs. SSLL executives specify that, even if their objective is to make free software a complete system of production, they do offer proprietary software to their customers. In addition, the "Open Source" movement can symbolize the hybridization of the systems

[20] In order to validate the statements of SSLL executives, a questionnaire is currently being prepared which should lead to a precise survey of SSLL employees.

[21] As the current putting into receivership of Alcôve demonstrates, even if liquidation is mainly the fruit of poor financial management, not of profitability (*Le journal du Net*, http://journaldunet.com).

[22] In this regard, we can highlight the fact that only one kind of SSLL type of English company was present at Birmingham's LinuxExpo this year, an event which is far less important its French equivalent.

[23] The European Commission advocates the use of free software in one of its reports (http://europa.eu.int/ISPO/ida/export/files/en/1115.pdf).

[24] To overcome this lack, Technopole de Soissons launched a contest of software creation in various fields like management and accounting: http://soissons-technopole.org.

of production themselves since it implies putting together a " free software core" with other software layers which are more specific and of a proprietary kind. IBM, for example, has implemented this type of strategy: it leaves the heart of the software program available to everybody and contributes to its improvement, while it protects the more specific software layers. This enables the company to stress its uniqueness and make money off it.

It is not yet certain if the free software industry is only in an emerging phase or if it will impose itself. There is a significant chance that it could lead to a hybridization of varied software models. Observing the SSLLs' activities, however, suggests the implementation of a new system of production. This system might not be incompatible with the old one. On the contrary, it may even complement it insofar as it allows the user to choose what he/she wants as well as to maintain diversity in an industry which is threatened by monopolies.

Company strategies for the freeing of a software source code: opportunities and difficulties. Lessons learned from two recent cases in the areas of CAD and digital simulation.

FRANÇOIS HORN,
University Charles de Gaulle of Lille III, CLERSE - IFRESI CNRS, horn@univ-lille3.fr.

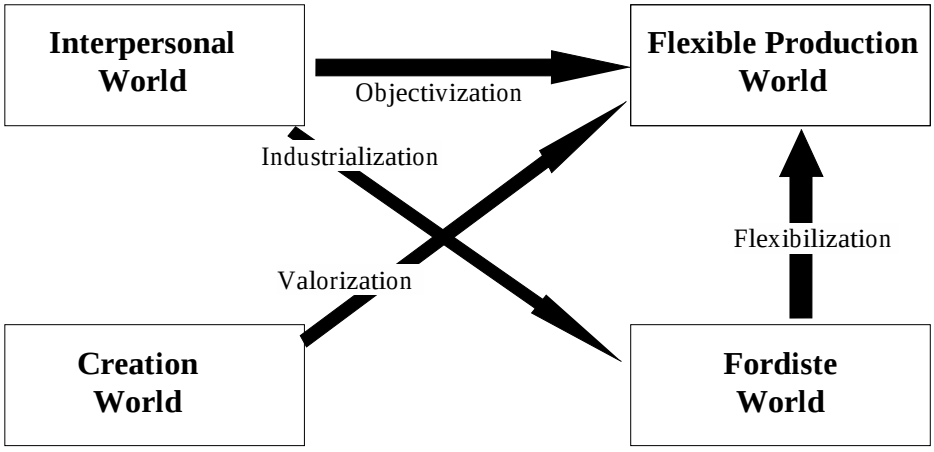
In spite of the undeniable progress that has been made in the field of software production, problems still remain. More specifically, these difficulties concern the realization of software which is not only highly dependable and well adapted to user needs, but which can also achieve high levels of productivity. Nevertheless, overcoming problems in one of these critical areas can often make progress in other areas virtually impossible. The consequence is that there are parallel processes of activity rationalization each of which is distinct according to the compromises that must be made for each activity. In the end, this is what makes software economics so diverse.

This diversity has been analyzed in terms of *worlds of software* production (Horn [2000b]). Each *production world* occupies a specific position in relation to some critical questions involving software production (cf. Table 1). The *interpersonal world* represents difficulties in conciliating productivity with high-level dependability in the production of custom software. The *Fordist world* of commercial software packages is characterized by high productivity but, all too often, by mediocre quality, and can only meet standard needs.

On the other hand, the *world of creation* of free software brings a certain amount of originality and undeniable efficiency to certain segments of the software economy. However, it has more difficulty extending itself to all its users and products. The software *world of flexible production* has the potential of reconciling productivity and adaptation to the needs of its users. Despite this, it has had a hard time making a place for itself, particularly under its most promising form, namely, custom software production based on standard components.

The coexistence of different *production worlds* is an on-going phenomenon in the software economy, most particularly because they complement each other, and this economy is evolving in a very dynamic manner. Changes can be seen in a large variety of ways and correspond to the shifts from one *production world* to another (cf. Figure 1). If industrialization leads to the mass production of standard goods (reflecting the *Fordist world* specific to commercial software packages), it also leads to greater flexibility and objectivity including the development of a *flexible software world of production* based on the three other *production worlds*.

Figure 7.1 — The dynamic Evolution of the Software Economy.



This analysis has made it possible to point out the current domination of the United States in the field of software economics, owing to its ability to industrialize its production much faster than is the case elsewhere. However, the existence of other approaches, most especially what we call "valorization", could contribute to change this state of affairs. Despite the fact that it inevitably contributes to the maintenance of different software packages, the development of the *world of free software package* production could offer genuine

Table 7.1 — Production Worlds in Software Economics.

	Absence of standardization.			**Product and/or its components standardization.**	
	Interpersonnel World.			***Flexible production World.***	
Dedicated Products	Product type:	Custom software developed from scratch.		Product type:	Standard software and custom-made services. Custom sofware based on standard components.
	Characteristic examples:	Specifi c applications.		Characteristic examples:	Company management system.
	Main producers:	Internal IT departments, IT service companies.		Main producers:	IT service companies, consulting companies, linked to Producers of equipment or software packages.
	Main users:	Large compagnies, Administrations.		Main users:	Large and medium-sized companies, administrations.
	Productivity in the production of the software:	Weak, Contradictory with dependability.		Productivity in the production of the software:	Relatively high level.
	Dependability of the software:	Contradictory with productivity.		Dependability of the software:	Relatively high level.
	Adaptation to user needs:	Variable according to the level of intercomprehension between producers and users.		Adaptation to user needs:	Depends on the quality of service relations.
	Creation World			***Fordiste World***	
Generic products	Product type:	Free Software (source code).		Product type:	Commercial software packages (object code and limited services).
	Characteristic examples :	Linux operating system.		Characteristic examples:	spreadsheet.
	Main producers:	University, research centers, independant creators.		Main producers:	Software “editors”.
	Main users:	fi rstly IT specialists.		Main users:	Households, companies (including S&Ms), administrations.
	Productivity in the production of the software:	Variable.		Productivity in the production of the software:	High.
	Dependability of the software:	Possible rapid improvement (for software which is initially successful).		Dependability of the software:	Often insuffi cient.
	Adaptation to user needs:	Strong for the computer science community. More problematical for average uers.		Adaptation to user needs:	Only for standard use.

solutions which could promote the birth of large-scale "customized" production in software economics. This could be achieved thanks to the production of free software components. However, the ease with which things have moved ahead so far in this field may not continue indefinitely[1]. Indeed, unfavorable factors could eventually prevail.

First, it is unlikely that those participating in the *world of creation* will be motivated to develop free software since these activities are generally perceived to be less prestigious and exciting than creation activities which they have already been involved in[2]. This is also the perception concerning the development of products which are not destined for the primary users (office automation and management software, public libraries of software components for commercial applications...)

Secondly, the growth of free software widens the market for equipment and complementary services producers. But this expansion can lead to the increase of "maverick" behavior on the part of some who might seek to benefit from the existence of free software without actually taking part in its development. This behavior is due to a series of interdependent phenomena: a) the appearance on the scene of participants who do not share the free software "culture" (IBM for example) as opposed to service businesses which have been involved from the start; b) possible changes in the cultural values of these companies as a consequence of their success[3]; c) tough competition in this particular market; d) sales to more diversified users who are not aware of the stakes involving free software and for whom the price of the solution is a determining factor; e) higher competitiveness of businesses which blocks them from dealing with the costs of free software expansion.

Thirdly, due to the expansion of free software, over the long run, some software package editors may refuse to support this trend if the products that are developed happen to compete with their own (especially since they often are in a weaker market position)[4].

Finally, the threat that free software represents for the interests of some powerful, commercial software package producers, has provoked powerful reactions. This clearly shows up in the way they have been registering patents. They either prevent the development of free software, as in the case of Microsoft, or they mislead them by creating software which is free only in name, such as Sun.

Consequently, new software that would be immediately transformed into free software is not at all expected to be produced at a fast pace. On the other hand, *free software businesses may transform former "private" software into free software and put it on the market*[5]*. Does this correspond to the emergence of new economic models and, if so, how do these models fit into the more general picture of software economics?*

To answer these questions, we have studied two recent cases, Matra Datavison's Open Cascade and E.D.F.[6]'s Code Aster, which are successively presented in the first and second parts of this article. Are these

[1] Until now the State has not assisted in the development of free software, which is not to say that it has not played an important indirect role. This is especially true in the use of public infrastructures, the work civil servants and the diffusion of a cultural model which is favorable to development of free software (i.e. the model for public scientifi c research).

[2] The prestige of an activity is especially related to its innovative character: developing a programmable text editor in the Seventies (similar to "Emacs" produced by MIT) could contribute to the fame of its authors; currently, this is certainly no longer the case.

[3] For example, Red Hat Software, the principal Linux "distributor", was listed on the Nasdaq on August 11, 1999

[4] Without taking sides, one can imagine that the desire to stand up to Microsoft's hegemony may have played an important role in certain positions taken that could be modifi ed if Microsoft's position were weakened.

[5] The software that is not freed is often called "proprietary software". The term "proprietary", which is widely used in the computer fi eld, is not adequate in our view - we do not really see how software can be a "proprietor"! - and it would seem wise to speak of "private" software to designate software that is controlled and owned by a company.

[6] France's National Electrical Utility Group.

cases only marginal ones which are inevitably doomed to failure or do they foreshadow significant new developments in software economics? Let us place these cases into the framework of software economics as analyzed through the prism of our *production worlds*. By so doing, we can benefit from the first lessons and draw some preliminary conclusions about the existence of new economic models and the opportunities these models can offer businesses. They also allow us to stress the difficulties encountered by businesses that want to adapt to new economic models (cf. Part 3).

7.1 Open Cascade.

Open Cascade software belongs to the C.F.A.O. sector (computer-assisted design and manufacturing). This sector plays a rather minor role in the software economy but continues to gain strategic importance for the companies that use it. Open Cascade is the first significant example of free software in this sector. In the same way, it is the first time that an important industrial group (E.A.D.S.[7]) has chosen to transform a commercial software package into free software.

Before examining the motivations that led to this decision and the investments that were necessary (1.2), and before we analyze the preliminary results of such a strategy (1.3), we must briefly trace the history of this experiment (1.1).

7.1.1 The historical background of Open Cascade

Qu'est-ce que Open Cascade?

Open Cascade is a library of software components (written in C++) which is used for the development of scientific and technical applications. In 1996, it included 20 000 C++ classes with, for example, forty-three different ways of finding the center of a circle. At the heart of this library is a 3D geometric modeler which comes with data exchanges and visualization libraries. It also has a tool that allows the development, configuration and management of attributes based on the geometric model.

Cascade is the fruit of the hard work of thirty people from Matra Datavision's R&D department and represents an investment of eight million euros. At the very beginning, these software tools were the heart of a development platform intended for the elaboration of the CAD Quantum software package which was to follow Euclid in Matra Datavision's offer.

From Euclid to the Quantum project.

About ten years ago, Euclid held a special place on the French CAD software package market. Unfortunately, Matra Datavision, its editor, regularly faced a very common syndrome that big software package developers had to deal with at the time, namely, the "spaghetti" syndrome. By constantly adding modifications, everything got jumbled up and it became impossible to change anything without modifying the ensemble in an unpredictable way. Most of its developers were devoted to the maintenance of a tool whose background no one could remember.

[7] Owner, among others of the company "Airbus Industry".

In order to prepare a new generation of EUCLID (named Quantum) which would break this trend, Matra Datavision decided to develop what was called, at the time, a "software engineering department" (CASE) and that is described as "middleware" today.

The objective was to develop CAD software intended for scientific and technical applications which would react vigorously to market trends and customer expectations. The latter needed to calculate more and more complex 3D models. To visualize the behavior of a satellite, for example, it is necessary to be competent in the field of thermics, electricity, radiation, etc. In order to open the various technological fields, Matra Datavision had imagined developing a "base" on which interlinked modules would fit. The idea proved to be excellent and was soon taken up by the company's major competitors.

After significant development efforts, Cascade, which Unix had promoted in the context of a 'proprietor' model and which was composed of a great number of tools and reusable components, Cascade started a double life. On the one hand, Cascade was the heart of the EUCLID software (and its future Quantum version) and made up the foundation on which it would be built. In addition, Matra Datavision also used it for a series of specific application projects. The company acted as the project leader and began marketing it in 1995 under the name CAS.CADE. Its purchasers used it on a large scale to develop CAD applications for themselves or for their customers. Cascade had approximately 130 active customers.

The Failure of the Quantum Project.

Unfortunately, the Quantum project was doomed to failure. This was due to the evolution of the sector which, as in all areas of software economics, was characterized by concentration on a world-wide scale. This was due to the strength of increasing adoption returns which more or less condemned average editors (Matra Datavision represents approximately 5 % of the world market for computer-assisted design). Moreover, the existence of simple graphic editors under Windows made it more difficult for Small and Medium-sized businesses to have access to the market. In the mid nineties, Matra Datavision made a marketing error by announcing the launching of the new Quantum version too soon. The latter was delayed because of the desire to integrate more and more functionalities thus prompting a wait-and-see attitude on the part of its customers. The requirements needed by such a product and the number of engineers necessary for its development kept growing whereas potential profit margins and markets continued to decrease. This does not even take into account the efforts required to develop such a distribution network.

In 1998, Hugues Rougier, Matra Datavision's new C.E.O, declared, "From now on, the future is in services". He also planned that the market for CAD software was going to follow the same trend that P.A.O (D.T.P.) software had ten years before and with functionalities that were ten times cheaper. Consequently, at the end of 1998, Matra Datavision decided to do away with the editor's economic model to refocus on service supplies based on one of its competitor's software packages, Dassault Systèmes' Catia. As a consequence, 150 employees lost their jobs while another 150 were hired in their stead. Today, Matra Datavision has more than 700 engineers and consultants working in Europe and the United States in the following fields: auditing and expert consulting, implementation, on-site and on-line training, engineering and specific development.

The company is ISO 9001-certified for the design and realization of services in engineering. It became IBM's key trading partner for the marketing of Catia and the world leader in the sales of Catia's new fifth version.

7.1.2 Decision-making and necessary investments.

Decision-making justification.

In this rather unremarkable context -other companies had made similar switches in the past- one question remained: what was Cascade's future?

Market research showed that Cascade users were very happy with it but, considering Matra Datavision's evolution, were also concerned about its long-term availability. Cascade required huge investments and guarantees on how they would be spent. Nevertheless, its commercial development was not adequate: approximately, FRF10M (€1,5M) sales turnover. The price of its license had to be dropped from 1 million francs to 250,000 francs (€38,000) and then finally to 100,000 francs (€15,000).

Matra Datavision was convinced that editors and integrators required such different skills that they could not coexist within the same structure. However, since it wanted Cascade to survive, Matra Datavision decided to transform it into free software while taking into account its social dimension. It is in this context that Open Cascade was born. Free software and companies evolving around such activities, such as MandrakeSoft, developed within this new context. A strategic study led by Matra Datavision confirmed this trend. After having thought of giving away only specific parts of its software, Matra Datavision decided to free the software (except for some advanced components ensuring the connection with other applications and the test base). Matra Datavision feared a negative reaction on behalf of the free software community since it needed its support. By the same token, Matra Datavision chose a type of license which implied that it would not ask for royalties for any developments stemming from Open Cascade. The goal was to reach a de facto standard. Indeed, Matra Datavision's objective was to compensate for the fact that it was not marketing its software by selling services linked to Open Cascade (on-site and on-line training, technical assistance, expert consulting, specific developments, etc.). It took until July 1999 to come to a final decision.

Improving the product

Before announcing that Open Cascade would enter the market free-of-cost in December 1999, they still had to spend more money so that the software package could run under Linux in order to reinforce its modularity and to improve its ergonomy. In short, Open Cascade simply had to "survive on its own". The non-paying users would only really utilize the system if they mastered it quickly. It is in this way that the work on Open Cascade architecture was undertaken. The documentation was completed and the programs written for all the main functions of the software. Above all, a very powerful module which reflected a great amount of work, was added to the platform. This led to the creation of the Application Framework Module (a deployment server managing the application architecture) which furnished everything that was necessary to build a complete application rapidly, including the man-machine interface, visualization and the saving of data.

The subsidiarisation of Open Cascade

The launching of Open Cascade was managed like any other project and included protection against possible leaks. Matra Datavision wanted to benefit from being the first company to offer a free 3D modeling software tool and feared that a competitor would snatch the idea before them. In March 2000, an initial business plan was launched. In fact, in addition to the initial investments that were necessary to improve the software, other expenditures were also necessary. This included an evening dedicated to the launching of Open Cascade, commercial work assisting clients who had purchased CAS.CADE, the creation of two Web sites (one .com destined to make the offer known and, especially, which permitted the downloading of the software and

a .org to serve the needs of Open Cascade users). Furthermore, a special CRM (Customer Relationship Management) tool was developed allowing them to manage potential clients among those persons who would download the software. New versions of Open Cascade were developed (after the development of a version for Linux and one for Windows, based on the 3.0 version certified in May 2000, on the November 2000 3.1 version and, finally, on the 4.0 version). The total amount of these investments can be estimated at about €800,000 for marketing, €200 to 300,000 for the development of the 4.0 version, €200,000 for miscellaneous services, and €100,000 for the development the CRM tool.

Initially, Open Cascade was a department of Matra Datavision. However, Matra Datavision, which had become a service company, no longer wanted to continue investing in this activity and sought to find outside financing by having Open Cascade break into a second market. Unfortunately, the collapse of the financial markets in September 2000, combined with the exorbitant demands of certain would-be partners (banks, capital venture companies) led to the progressive abandonment of the project. Moreover, the closing of one of Matra Datavision's departments required the opinion of the Workers' Council at a time when labor relations were conflictual as a result of negotiations concerning the 35-hour work week. In the end, it was only after one year that Open Cascade was subsidiarised under its own name with 100 % of the capital being maintained by Matra Datavision.

7.1.3 Preliminary results.

Success in terms of diffusion.

Without doubt, the transformation of Cascade into free software was very successful regarding its diffusion and this gave confidence to the users that the product would be around for a long time. Open Cascade found its place at a time when there were only three such platforms in the world. "We didn't think that we would have been so successful" confided Hugues Rougier, Matra Datavision's CEO. After the hesitations concerning its simultaneous launching in the United States and Europe, the decision had been taken to concentrate only on Europe.

After the first year, twelve thousand users from seventy-four countries had downloaded the source code of the software package and over 30,000 people visited the Web sites per month. According to the marketing director, the number of Cascade users went from 100 to 2000 from the time it had become a free software: 70 % of the users are editors, who often require only a small part of the software to develop specific CAD applications. Major companies chose Open Cascade to develop their own applications. For example, the CEA created its own nuclear simulation tools in a 3D environment using Open Cascade. Likewise, Honda followed suit to design its motorcycles and Daimler-Chrysler used it to test automobile parts. For veteran users of Cas.Cade, the opening of a source code offered advantages. As a consequence, Optis, a CAD editor specialized in optical simulation software, chose Cascade (before it was freed) to develop its own Speos software. One Optis spokesman states that, since the time Cas.Cade has been opened, "the number of its users has multiplied and, at the same time, the number of questions asked in the Open Cascade forum also has also risen dramatically. Therefore, we immediately know when a problem has been identified and, in this way, it can be solved much faster." (*Le Monde Informatique*, 25/01/2002).

The advanced use of Open Cascade became the subject of a course at the Léonard de Vinci University in France and at Berkeley in the United States, which selected Open Cascade to develop its CAD applications and decided to continue to enrich the software (i.e. the publication of components destined to optimize the triangulation of surfaces and a new user-interface). Other partnerships were begun with the Universities of Karlsruhe, Parme and Tel Aviv, and the number of university users is estimated at about 250. This recognition

from the university community is important because it allows Open Cascade to be enriched by the research of these institutions and it also guarantees that employers will be able to recruit engineers that have already been trained in the use of Open Cascade.

The management of service activities.

At the same time, the sale of services involving Open Cascade climbed considerably and the turnover from the sale of services for 2000 alone rose by 55 % as compared to the turnover for 1999 which also included the sale of licenses. The sales network relies on fifty commercial engineers at Matra Datavision who have Catia 4,000 clients. Open Cascade has also reinforced its internal team by increasing the number of its marketing specialists trained in the use of Open Cascade from two to seven. It has also developed tools which monitor the behavior of people who consult the sites as well as the management of a data base containing 15,000 contacts.

Initially, Open Cascade commercialized its services under the form of three packages (Silver, Gold, Platinum) which included combinations of miscellaneous services (helpdesk, e-learning, on-site and on-line training, technical assistance, expert consulting, development management, personalized follow-up by a project manager). Afterwards, Open Cascade innovated by proposing three new *à la carte* offers of support (Advantage, Premium, Excellence). Each offer is composed of a certain number of tokens during the trimester and the user can dispose of them freely on the services they need (each service being worth a certain number of tokens).

The economic model.

The economic model is based on a new approach whereby it is the "know-how surrounding the code rather than the code itself which brings in the profits" (Abou-Haidar, marketing director). It is based on a distinction between three activities: the development of the software itself, the typical kinds of services used by editors involving software (training, integration, maintenance) and the specific developments that have been made thanks to this software. As the software is provided for free, the last two activities must finance the first one. The reasoning is as follows: the freeing of the source code of the software must a) allow for the continuous development of a product at a lower cost and b) make it profitable thanks to its favorable position with regard to associated services.

a) Continuous development of a quality product at a lower cost.

Being a free software ought to permit the user to improve the software and develop new functionnalities. This can be accomplished thanks to the availability of a free platform to create specific applications as well as the possibility of changing them and/or expanding them by adding new components and new functions. The consequence of all this is that a large group of software specialists are going to enlarge and complement the team of internal developers. Open Cascade thus furnishes certified versions at specific intervals to a community of international developers. With this in mind, a pilot committee was created. It is an international team associating the main industrial and university users. Their role is to apply the development projects through a network made up of developers from inside and outside Matra Datavision.

In the year 2000, fifty outside contributors to Open Cascade provided various kinds of assistance: transferring software to other systems (IRIX 64 bits, Alpha OSF), correcting defects (memory leaks...) and translating the tutorial into Spanish, etc. Currently, there are seventy active contributors and the objective is to reach one hundred. These outside contributions are significant. Open Cascade estimates that they represent about 20 %

of the value of the software. According to *Industries et Techniques* (February 2001), Matra Datavision had to inject approximately 2 million euros per year to continue to develop its tools. In 2000, the company limited the costs to €1.2 million. Nevertheless, these contributions occur especially at the periphery of the software, not at its center. They enrich it but do not consolidate the platform and its development is still expensive for Open Cascade (about a 1 million euros per year). Nevertheless, this cost has been minimized by the hiring of Russian computer scientists. Today, the company is made up of one hundred and ten people of whom seventy are Russian (at Nizhny Novgorod), thirty-eight are French and two are Japanese. The cost price in France for a project leader is €650 per day, €450 for a developer, whereas in Russia it is slightly above €100.

b) What profits can be made from associated services?

Job possibilities in terms of services. There are two big markets for Open Cascade: the first involves editing services and the second is for specific developments.

The first category includes training, maintenance and integration. The latter is becoming increasingly important with the digital continuity in PLM (Product Life Cycle Management). This makes the conception of the product the basis of a company's management system. Open Cascade possesses assets which allow it to integrate other software since the source code is public and, not being dominant on the market, the interface capacities are not particularly developed. From this point of view, the use of this software guarantees a certain independence with regard to the other CAD tools. At the present time, these activities bring in a few million € which do not cover the expenses. However, efficient tracing of users has only been undertaken since the end of the year 2000. This should spur some extra business activity.

The second market concerns specific developments which have been undertaken for software editors (i.e. Samtech which has commercialized solutions for the calculation of structures) or for industry. An example is Mitutoyo, one of the three biggest measuring machine manufacturers which an application permitting them to confront digital models and real models through the integration of tolerance measurements). For specific developments, the activity has proved itself to be very lucrative with profit margins superior to those of Matra Datavision.

A marketing tool for potential clients who are prepared to spend more. As for the sale of these services, the free character of the software can be regarded as a marketing tool in and of itself. The development costs of the software can then be compared to promotion costs. Hugues Rougier states that, "We have to compensate with complementary activities, just as in the case of a supermarket which sells off certain products in order to compensate for losses in other areas." The costs can also be assimilated to sponsoring. Rougier adds that this strategy permits a software which does not have a dominant market share to achieve this goal.) Open Cascade was the subject of one hundred and thirty articles whereas, with a bigger communications budget, Matra Datavision only has around sixty per month.

The free distribution of the certified version boosted confidence in its diffusion and significantly increased the number of potential clients. "The users come to us," declared Pierre Bruno, Open Cascade's CEO. The freeing of the software is an efficient means to accelerate access to the client with an additional benefit being that the contacts in companies are established with the very people who are interested in the product. Yet, the commercial costs before the sale can be very high. For Matra Datavision these costs exceed 10 million euros annually. Furthermore, since user companies do not have to pay for the license to acquire the software, they can spend more on services. Cirtes, a company that manufactures orthopedic corsets and which uses Open

Cascade to develop modeling software, has indicated that "downloading software free-of-cost gives the user the impression that everything is free. But one must not forget that the value of this kind of development tool comes, above all else, from the support that surrounds it." (*Industries et Techniques*, February 2001). This being the case, according to Open Cascade, "the application of open source solutions, including services, is 40 % cheaper than those of proprietary products". The question for Open Cascade remains whether it can succeed in transforming some of its users into clients of the services offered by the company.

Advantages in relation to the competition. In the case of free software, nothing stops the company which does not participate in the development of free software (or which only does so marginally) from selling services linked to this software. The experience is very recent indeed. Open Cascade has only limited competition with regard to the services it proposes. On the other hand, some companies, most notably Indian ones, are utilizing Open Cascade to develop cheap CAD software which could eventually compete with Catia for entry-level projects.

With regard to services proposed by Open Cascade, the general feeling is that there is nothing to be feared from companies which might be more competitive because they do not have the development costs of the software they use. Hugues Rougier estimates that companies will prefer referring to Open Cascade's development specialists. He compares the situation to those of consulting groups which make their methods public. Although this allows them to be used by competing companies, the fact of being the author of a given method brings confidence and security to those offering the service. Furthermore, Open Cascade considers that it is several years ahead in the area of the commercial management of its new activity, namely, proposing services to clients who have downloaded free software. They are skilled in the subtle analysis of their customers' needs and in being able to make focussed propositions rapidly. They have also gained expertise in the contractual management of the customer, in distance management as well as in the management of offshore resources. Finally, they have developed efficient tools, most notably through the development of a CRM software.

Difficulties encountered by Open Cascade.

Nevertheless, Open Cascade did encounter some difficulties in the implementation of this economic model. These difficulties definitely show in the company's current results and short-term forecast: in 2000, it had a €2.1M turnover with a €2.3M loss, in 2001 it had a €3.5M turnover with a €2.0M loss. In 2002, the company's objective is to reach a €4.2 turnover with and a €1.0M loss and for 2003 its goal is to balance its budget with a €6.0M turnover.

Some of these difficulties may indeed be linked to the present economic climate: in 2000, the company had to revise some of its investments downwards in the field of information and communication processing due to the evolution of the economic situation. This led to the postponing or simple cancellation of major projects[8]. Such measures could have significant consequences on Open Cascade since it deals with highly technical contracts in which the decision-making process is exceptionally long.

On the other hand, Open Cascade is fully aware of some of its initial mistakes. The most damaging consequences resulted from the fact that the people who downloaded the software could not be identified. This stems from the Internet's initial ethical guidelines, Open Cascade was reluctant to ask for those people's email.

The biggest difficulty came from the real cultural changes deriving from mutations in the economic model.

[8] In 2001, according to Gartner-Dataquest, the budget for equipment in the TICs (computers, servers, warehousing, software, telecoms) was reduced by 8.4 % (the first reduction since 1958) whereas it had progressed by 10 % to 15 % per year during the 1990s

This is particularly true for a company that is used to a certain passivity due to its past. It is not that easy to transform a traditional R&D department into a dynamic business. Indeed, the success of the company presupposes that the transformation of tens of thousands of users, over the medium term, into 200 customers. Unfortunately, the company only managed to go from 17 customers in 2000 to 30 in 2001. Its present objective, which has not yet been reached, is to reach 70 customers. If the freeing of the source code has made it possible to increase the number of its users well beyond the initial forecast, it appears much harder to make these users actually buy services. They easily get into the habit of using the software for free, but find it difficult to have to pay for its services. For the most common ones - help desks, for example- there even exists a relative contradiction between these objectives: on the one hand, to increase the diffusion of the software and, on the other hand, to attract customers. Indeed, the success of the software diffusion entails that it can be used easily. This goal was achieved thanks to its significant improvement and discussion forums that have worked really well. As a consequence, unless the company offers first-class, but also very expensive services that give the users quick and very precise answers, they do not see the point of buying them. Right now, these difficulties are pushing the company to think about selling certified versions of the software. The versions which are currently being developed can still be downloaded, free of charge.

7.2 The freeing of E.D.F.'s Code Aster Source Code.

The following case, which differs considerably from the previous one, as far as its set goals are concerned, is even more recent. Before we examine its preliminary results (B), it is necessary to review the history of Code Aster in order to understand why E.D.F. chose to transform this software into a free software program (A).

7.2.1 The history of Code Aster.

Once we have explained what Code Aster is all about (1), we will see how E.D.F.'s failure to try to commercialize this software (2) encouraged the company to transform it into a free software program (3).

What is Code_Aster?

Code_Aster is a digital simulation software program which makes it easier for E.D.F. to maintain its production installations and transport its electricity. It was conceived to answer the group's needs in the field of mechanical analysis and multi-physical simulation. Indeed, E.D.F.'s strictness in terms of security and availability requires digital simulation backup when decisions need to be taken concerning the working of, repairs or replacement of its equipment. One of this programs main functions is to evaluate the wear and tear on equipment used for the distribution and production of electricity. This also includes E.D.F.'s nuclear installations.

E.D.F. runs extremely diverse projects (260.000 line supports, 300 turbo alternator groups, 213 dams, 58 nuclear power plants) and uses highly variable kinds of equipment. This software program makes it possible to deal with a vast number of simulations: thermo-mechanics, linear and non-linear 3D, static and dynamic, boiler-making industry, civil engineering and the complex geometry of porous terrains.

E.D.F.'s development and diffusion process is based on a level of quality that has been well proved and verified. Indeed, the Code Aster program's quality requirements are particularly high (the same quality as required in the field of "nuclear engineering"). This is due to the fact that they must justify the life span of their installations to the Security Authorities. E.D.F.'s R&D Department has been developing this software program for the last 12 years and a dozen developers spend most of their time working on it. It includes

800,000 configured code lines, 1,170 non-regression and qualification tests and 9,000 pages of documentation (5 manuals in 9 volumes). 190 people use it regularly, around forty of whom are industrialists or university researchers who do not work for the company. In 2000, E.D.F. itself spent 35,000 calculation hours, including 16 daily calculations that took over 2 hours each to complete.

The failure of the commercial solutions.

Code Aster is one of the consequences of E.D.F.'s ambitious policy to master digital simulation in the different fields deriving from its main activity, that is, producing and distributing electricity: fluid mechanics, neutrons (fast-breeder reactors) and solid mechanics. In 1988, this ambitious approach led to the development of a single code for structure mechanics: Code Aster. This effort to invest in a software program is one of E.D.F.'s identifying characteristics: the company is one of the last European groups to make such investments. The rest buy their tools directly from the United States.

E.D.F. quickly concluded that it was in its own interest to capitalize on such an investment but found it difficult to come up with an adequate commercial sales technique for its software. Indeed, E.D.F. could not become a software editor since its status, under French law, does not allow the company to develop certain activities (for example, computer services). In addition, the peripheral nature of its activities stops it from being a legitimate software editor: an R&D team is very different from one which deals with customer assistance. Some experiences in company creation to market their software abroad ended in failure, for example: Simulog, part of INRIA, which deals with thermohydraulics and fluid mechanics.

Nevertheless, E.D.F. tried to sell some Code Aster licenses (€16.7k each), indirectly, by using three distributors and service providers with the hope of creating a synergy with these companies: CS (ex CISI) a scientific software program editor aiming particularly at S&Ms, CETIM which would be the technical guarantor for mechanical industries specializing in scientific domains and, finally, SAMTEC, a Belgian company which editing software programs and providing services in the field of aeronautics and space research. However, Code Aster had a weak point: it did not have any pre-treatment or post-treatment modules which allowed the data transfer necessary for any type of digital simulation and CAD. In this field, Samtec's strategy was to rely on Cascade. The plan had been to utilize a development mutualization program, using Cascade as a starting point, to enrich the software. Matra Datavision's December 1998 decision to end its software editing activities and to transform the company into a Catia distributor and offer its services in support of this software, led to considerable uncertainty with regard to Cascade's future and the abandoning of the project. E.D.F. tried to find less ambitious solutions. The conflicting strategies of the different partners were all but impossible to reconcile and this stopped the projects from being realized.

In 1999, E.D.F. found itself in a highly awkward situation. The distribution mechanisms did not function well, there were problems dealing with customers who were turning to them because this was not E.D.F.'s job and, finally, license sales were not bringing in any profits. It is in this context that Matra Datavision made the decision to transform Cascade into free software, a choice which had considerable consequences on E.D.F.'s decision.

The decision-making process.

In September 1999, E.D.F. came back on its decision to invest in the development of pre-treatment or post-treatment modules, considering that it was not its responsibility to do so and hoping someone else would come up with the solutions. This was clearly interpreted as meaning that they were also renouncing on the commercial diffusion of Code Aster. At the end of November, a pilot committee decided to give the green

light for the diffusion of the free software. In July 2000, after a thorough review of the commercial sector and an assessment of its distributors, E.D.F. decided to terminate the distribution and to no longer adhere to the commercial model and to do an in-depth study of the free software model, including an evaluation of risks, legal aspects and the license choice. They also decided to study the creation of Internet sites, the economic pertinence of the model and the opportunity to be the first to propose, within the free software framework, a complete simulation offer. Finally, they sought to examine the possibility of finding outside contributors. This implied convincing a number of industrialists and technical centers to participate in the creation of an external pilot committee.

The decision was taken to create web sites on the Code Aster software, along with the use of Intranet which would help to improve the flow information from within E.D.F.. In addition to this, the Extranet for E.D.F. suppliers who were using the software would be made available as well as the use of an Internet site which would function independently of the official E.D.F. site. The investment would amount to only $\frac{1}{2}$ a man per year although further investment would be necessary to create English and Spanish versions of the site.

The participants who would be able to compose a team of outside contributors rested on a few interested partners (Institut Français du Pétrole, Michelin and IFP, a company which commercializes software and services). However, constant changes in strategy by these partners in a very unstable environment ruined the chances of a simultaneous accord between them. This led to the temporary abandonment of the idea of an inside group of participants.

In March 2001, the Internet site was operational. After discussions with members of the free software community and with Open Cascade, the decision-making projects and the license projects were ready with the legal backup. The license that was chosen was the GPL (GNU General Public License) certified by the Free Software Foundation which allows users to develop complementary codes, to build specific commercial applications and to propose services related to the software. The different E.D.F. actors were convinced of the soundness of the project and the decision was taken on July 6 to move ahead quickly in order to capitalize on the fact they were the first in the field. On October 19, E.D.F. made Code-Aster free software available on its Internet site.

7.2.2 Preliminary results.

Success in terms of diffusion.

100 days after the opening of the site, it had been visited by 14,000 people of whom 65 % were from France, 39 % from the united States - these connections were generally very brief given that the site is exclusively in French - and 6 % from 32 other countries. There were 440 executable downloads (87 % from the Linux version, 13 % from the Solaris Irix version, and it took 2 months to have a request for Windows NT version). Only 50 % of the downloaders were actually identified since user identification is optional. Among the identified downloaders, it turns out that 60 % were academics and 40 % from industry, and 83 % were from France, the rest divided between 20 other countries. The source code was not put online until three months later and was also massively loaded (140 downloads in 40 days). At first, the project initiators were somewhat worried about the exchanges but they turned out to be of good quality. Although the answers are essentially the work of E.D.F. employees, the different participants in the project also contribute to offer quick and spontaneous responses.

This success was due in part to the publicity that the "free community" provided E.D.F. regarding its

decision. According to Jean-Raymond Lévesque, the initiator of the project, "We benefited greatly from the fantastic help provided by the advocates of free software." The most notable assistance came from young computer scientists working in industrial companies.

Increasing demand for more services is becoming increasingly apparent but, in this domain, E.D.F.'s objectives are far different from those of Open Cascade.

Different objectives.

E.D.F.'s objective is to improve the software in order to better respond to its internal needs. The external diffusion of Code_Aster via the Internet under the form of "free software" ought to help expand the network of users. The huge mass of very dedicated specialists who are using the software should also serve to test it and will consequently be able to back up the opinions of experts who have been hired to validate the code before it is put into production. In fact, according Jean-Raymond Lévesque, "the value of a software simulation and study program rests largely on the competence and the critical spirit of its users[9]." The objective is to obtain "recognition through use" of the software. This approach increases the chances of detecting anomalies and allows for the comparison with similar existing products as well as the validation of the software by the users. In the latter case, the evaluation of risks is generating an increasing requirement for openness: "The validation of the new functions of Code_Aster, on account of the strict requirements of the nuclear sector, is difficult when the software is only used internally. In fact, we do not have enough users to qualify the new developments quickly." (Jean-Raymond Lévesque, *Le Monde Informatique*, 1/2/2002). This incremental improvement of the software, which is facilitated by its cost-free availability, is decisive for a highly technical software where the diffusion time is generally extremely long - a digital simulation software developed 12 years ago is considered to be a "new" product - and where attempts to totally renovate the architecture are very difficult (i.e. an attempt by Berkley to design revolutionary architecture ended in failure).

In order to accomplish this goal, it is necessary to establish a strong link between the increasing quality of the product and the involvement of the users. The logic behind this is that, in the end, everyone ends up a winner. The increase in the number of users can occur in several domains thanks to the fact that the software is free. E.D.F.'s suppliers, who are confronted with a purchasing policy which is getting stricter as well as a steady increase in demands without any market guarantees, combined with pressure on account of prices, benefit from having a free Code Aster user license (exploitation and development) to undertake work for third parties. This opportunity can also benefit other industrialists and digital simulation services suppliers. This encourages external users to provide their opinions of software functions required outside the context of E.D.F.. A second axis of development concerns research teams and development agencies which might be interested in the free offer of a 3D simulation platform with several entry points. These are independent of commercial solutions and, most notably, permit the testing of the development version to proceed more quickly. Free access to the software, to its theoretical documentation and to a corpus of tests and examples can also facilitate the use of Code Aster in training contexts. All of this leads to the steady development of a pool of trained participants, a fact which is a crucial asset in the future diffusion of the software.

If E.D.F.'s principal goal remains the qualitative improvement of the software thanks to the broad experience of its users, E.D.F. also hopes to enrich its software through the contributions of a large community of developers. According to Jean-Luc Dormoy, "The objective is to lower the computing costs while simultaneously encouraging co-operation between several companies." (*Le Monde Informatique*, 1/2/2002). E.D.F. hopes that this approach will help it to capitalize on the contributions made by numerous university or industrial research teams and, in return, offer them permanent reception facilities which can be easily used

[9] Personal communication.

for their own work. "We hope that the number of contributors will double by next year. This collaboration within a well-structured framework will bring about much higher quality than the efforts of a single isolated team" says Jean-Raymond Lévesque (*Les Echos*, 19/11/2001).

Nevertheless, one important difference with Open Cascade is that E.D.F. has no intention of commercializing the services involving Code Aster. "We want to see a group of services emerge which are independent of us" asserts Jean-Raymond Lévesque. When E.D.F. publicly announced that Code Aster would be transformed into free software, it specified that "any organization can propose conditions on the use of the software which do not require the approval of E.D.F. (training, support in the broad sense of the term, home-delivery, qualification) and for the enrichment of the software (training, support, special distribution)." Since they were convinced that there is strong synergy between the diffusion of the software and the creation of a network of service companies around Code_Aster, E.D.F. helped in developing these services by proposing, along with the authors of the software, free training in the use and development of the software as well as telephone assistance to the interested organizations.

The Perspectives.

Thanks to an initial surge of curiosity, visits to the sites have been growing steadily, especially in the industrialized world. Courses on Code Aster are scheduled for next year. The E.D.F. developers' first assessment of their activities has been positive. In the words of one spokesman, "it works and it's useful". This most notably due to the use of Code Aster in previously unforeseen areas. E.D.F. is continuing its development for its own internal needs and intends to finish version 6 of the software by October 2002 and implement version 7 by October 2004. The company plans to continue developing networks in order to facilitate the outside contributions made by academics and industrialists. A General Assembly of users is planned for October 2002 in which there will be a "user" workshop to make an assessment of the various ways the software has been employed as well as their services and distribution networks. A "contribution" workshop will be organized, the goal of which will be the adoption of a good conduct charter and the creation of a pilot committee.

Over the long run, planners do not exclude the possibility that the product may drift outside of E.D.F.'s direct control, a situation which would lighten E.D.F.'s financial load while simultaneously protecting the product and jobs.

7.3 First lessons.

It is obvious that the limited number of cases, combined with the fact that they are recent, warns against drawing any final conclusions. It is possible, nonetheless, to point out a few lessons which can be compared to similar cases which have already been studied.

7.3.1 Constrained strategies.

The first lesson is that in the two cases, the choice to transform the software into free software resulted in commercial failure: the failure of the Quantum project and turning towards services in the case of Matra Datavision. In the case of E.D.F., the transformation of Code Aster into commercial product also resulted in failure. Nevertheless, these failures do not call the intrinsic qualities of these software programs into question. As we have already shown, in software economics the importance of increasing output, especially from

external networks, demonstrates that a solution which is not technically optimal can nevertheless dominate.

This feature of software economics is not always well understood. For example, the new director of E.D.F.'s R&D department to whom the Code Aster project was presented admitted that he could not understand the claim that it was "a valuable product despite the fact that one cannot do business with it." As Hugues Rougier (PDG de Matra Datavision) put it, in order for a strategy of transforming software into free software to be successful, "the product must be good, useful, dependable, and of excellent quality." The reputation of the software can be difficult to create but this does not imply that the product has any technical shortcomings. In a situation where a software program is confronted by a dominant company which controls the lion's share of the market, its transformation into free software appears to be a tactic of "the weak against the strong" (ibid.).

7.3.2 Software computer scientists: a layer model.

The second lesson to be considered is that free software must be used by computer scientists. Indeed, the distinctive characteristic of a free software program is that its source code is freely accessible. The free cost of the software is only an inevitable consequence of that fact. As Hugues Rougier has indicated, "Having access to the sources must serve a purpose." Yet, this access only has a direct interest for computer scientists. For non computer scientists, the knowledge of the source code is absolutely useless.

The kinds of software that can be interesting when using a free software approach concerns those which are also tools used by computer scientists. These can be designed to perform tasks such as digital simulation where opening the code serves to be adapted it for special purposes. Otherwise, it can be utilized to improve specific software as in the case of a development platform. We can add that these activities must not be at the heart of the company's primary functions. For example, the companies which use a digital simulation software tool, whose code is public, jealously protect the confidentiality and the private nature of the data on which their simulations are based because this is important to their competitiveness.

The proportion of software used by computer scientists is bound to increase with the extension of the "layer model". This model consists in pursuing the detour in software production. More and more software is being produced in conjunction with other software programs which are themselves created thanks to yet other software programs. The extreme consequence of this process would be software production based on component software which would facilitate the programming design for rapidly growing products.

This approach corresponds to one of the major stakes in software economics. Faced with the permanent extension of software use and the diversity of needs, the development of custom software based on standardized components (i.e. the second modality of the *flexible production world*) appears to be a solution which would allow for the reconciliation of productivity and relatively high reliability (through the reutilization of tested components) while at the same time being able to adapt the software to the precise needs of the user. But the development of this kind of particularly promising production cannot appear as if by magic.

To be totally efficient, component software must be chosen from all of those that have been produced by the computer science community. It is obvious that one cannot limit oneself to internally produced products. The problem is that the use of component software necessitates direct access to the source code of these components in order to solve problems of interoperability, reliability and durability. The existence of these components under the form of free component software can thus offer an interesting solution. The company which frees a "layer" of software "bricks" can finance this activity by adding another "layer" of software and/or

services in a domain where it can expect to be more competitive.

7.3.3 The Search for a Standard.

This evolution raises the question of standardization which, just as in the computer science field more generally, has been so significant for software development, especially given the intensity of technological interlinkage between the different software components and computer system equipment. From this point of view, transformation into free software can be an efficient means to impose a standard in order to become an indisputable reference in a given domain. For example, the fantastic development of Internet and of the World-Wide-Web is due to the success of standardization processes over open standards that depend on free software. Reciprocally, the existence of Internet plays a major role in the diffusion of free software as well as in its improvement and development.

In the examples that have been studied, a major preoccupation remains that the standards will be created around the freed software. This explains the will to be first in the this sector and the choice of certain decisions, in terms of licenses, for example, in order to benefit from the recognition of the free software community and to accelerate the standardization process. The success of these processes can condition the spread of a network either to improve the product (cf. E.D.F.) or to commercialize its services (cf. Open Cascade). At the same time, if the efficiency of this approach stems from its originality, certain decisions to free other software in sectors were free software already exists can be viewed as being more hypothetical. On the other hand, the interconnections in different fields might incite those working in the same area where free software already exists to free their own software.

It is not by chance that the industrial companies that have followed this path happen to be working in related domains. Hugues Rougier estimates that free software is destined to expand steadily.

7.3.4 Limited successes must not mask the difficulty of adapting oneself to unedited economic models.

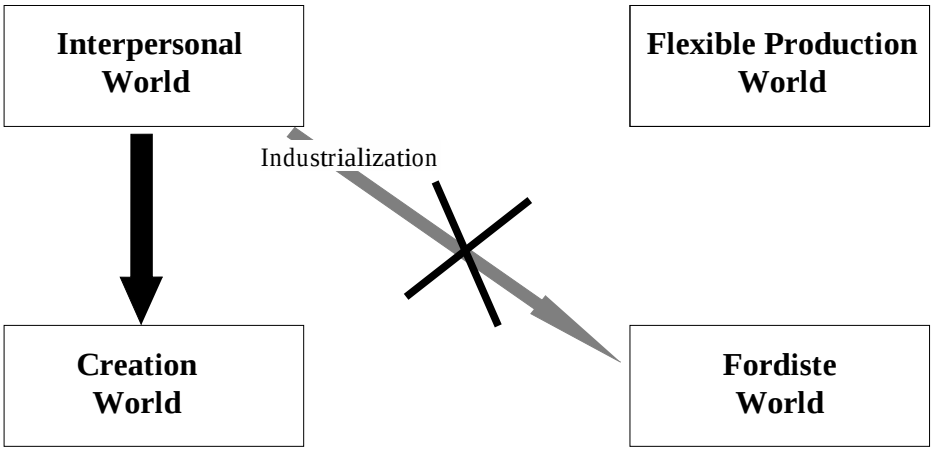
The principal success concerns the improvement of the quality of the freed software which is diffused quickly. Beyond the fact that it is free, the principal characteristic of a free software package is that the contents are made public. One can see how it is designed, how it is written, perceive errors and identify malfunctions with precision. Inversely, it is very hard to attribute problems encountered in the use of a global computer system to specific software without having the source code of the software programs which serve to interact with them. This explains why a company cannot free only high quality software and that, in the two cases studied, important investments were made to improve the software even more before publishing its source code. In fact, such decisions could affect the company image beyond a single free software company. This explains why certain companies, like Thomson's, which have large numbers of software programs that have lost prestige, are anxious, not to offer the software for free, but rather to make their code sources public. High quality software is diffused much more widely and benefits from the continuous contributions and improvements of its users.

The two cases studied here confirm this line of reasoning, but they also show the difficulties encountered in adopting radically new economic models. Before examining these difficulties, it is necessary to understand these two rather different models with the hope that greater insight may be gained in analyzing the other experiences involving the freeing of software.

The first of these models, illustrated by the E.D.F. experience, rests on the will to establish a strategic

product over the long term. In our representation of the dynamic evolution of software economics, this model corresponds to a company located in the *interpersonal world of production* - a software developed to meet internal needs - which does not manage to transform its software into a commercial software package (failure of the approach which we described as "industrialization"). In diagram II, the transformation into free software corresponds to a new approach which leaves the *interpersonal world* and moves towards the *creation world*, the objective being to continue improving the software qualitatively for the internal needs of the company. This is accomplished thanks to the feedback of users who are becoming more and more diversified. At the same time, external contributors help minimize the development costs. Nevertheless, the company's objective is not to reach the *flexible world of production* by the marketing of services and customized development based on free software (i.e. the approach called "valorization"). If such evolution in this direction is desirable to ensure the long-term survivability of the software, it will be due to the work of contributors from outside the company (with the possible assistance of the company). The company will not necessarily profit from it commercially. If, in the examples studied above, it is the strategic character of the software which is stressed, one can also imagine that such a model is relevant for strategic software at a nation-wide level. An example if this might be the goal to achieve technical independence for the control of software intended for low level defense activities. This raises the question of possible public financing for the adoption of this model.

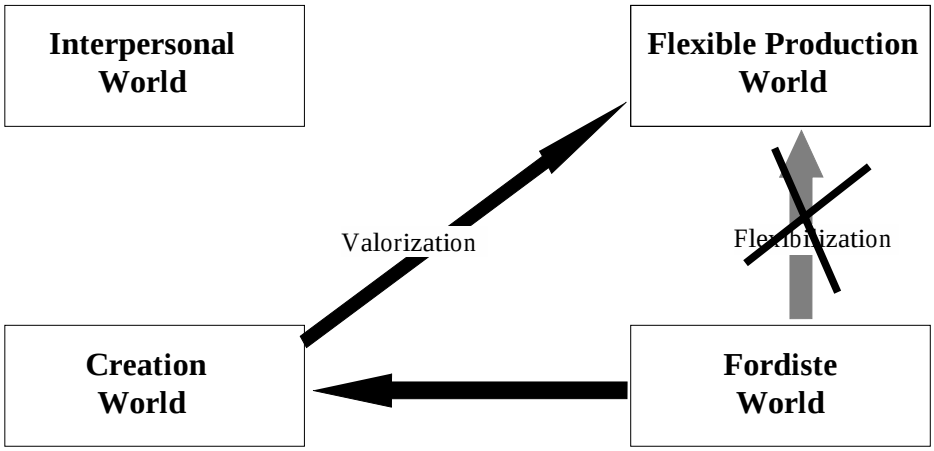
Figure 7.2 — A First Model: Extending the Life of a Strategic Product by Freeing It.



The second model corresponds to the Open Cascade experience and is based on the sale of various services related to free software. This includes custom-made development software. Here the objective is to find a niche in the *flexible production world* which, in software economics, is considered to be the future of any gainful activities. In diagram III, the starting point of this evolution is a company located in the "*Fordist world*" in the role of a software package editor. Confronted with its inability to develop these new activities - taking the example of a software package which is on the verge of commercial failure - difficulties to adapt to an approach known as "flexibilization" - the choice is made to evolve in the direction of the *world of creation* (i.e. transformation into free software) but with the ambition of benefiting from this position so that the company can reach the *flexible production world* while following the approach known as "valorization". In this case, the transformation into free software involves a marketing approach requiring an investment on the part of the company. By adopting this strategy, the company hopes to gain a competitive edge in prospecting for customers. It also aims to benefit from the quality of the solutions suggested for the services market tied to free software. The relevance of this model implies that there is indeed an important market for services that are complemented by free software. For example, Intelli CAD tried, without success, to follow this strategy by freeing its Visio software to compete with Autodesk in the CAD field for entry-level products. It was found, however, to be in a delicate situation because it had no real services to offer (*Industries and Techniques*,

February 2001).

Figure 7.3 — A second Model: Sales of Services Linked to Free Software.



Furthermore, the evolution of companies in the direction of these innovative economic models present difficulties.

The first of these difficulties lies in the cultural changes which this transformation represents for the company. First of all, these cultural changes relate to the way software is produced. Developers "have a hard time publishing their sources" explains Jean-Raymond Lévesque. There is a certain reticence to reveal the intimacy of their work and to publicly unveil the way in which they conceive and write the software. At the same time, they open themselves up to much greater outside criticism. "Will computer scientists write in the same way now that it is free?" asked the person in charge quality control for Code Aster. For computer scientists working within the framework of a company, producing free software constitutes a break with the past which is far more brutal than for university computer specialists. The latter have an advantage in that they played a major role in development of the first free software and, furthermore, they are accustomed to making their work public and being judged by their peers. As a consequence, the effectiveness of criticism and the higher level of requirements resulting from this, explain the superior quality of the free software as compared to commercial software. Indeed, the lack of transparency can lead to the masking of design errors which are extremely difficult to detect once a software program is integrated into a global computer system.

There are also substantial cultural changes in the nature of economic activity. For instance, in the case of E.D.F., it became imperative for each collaborator to show his worth - "You are not part of the CNRS!" is a frequently heard criticism - and it is not always easy for the developers or the managers to participate in an activity in which he appears to be a disinterested party. In the same way, Abou-Haidar explains that in the case of Open Cascade, "when we announced to them that their work was going to made available to the entire world free-of-cost, it is a little as if we had ripped one of their arms off. We had to make them aware that they now had to share their work. That was the hardest part." (*Linux +*, September 2000). Consequently, adapting to this environment is not easy for certain employees whose behavior has been conditioned by years of work in an R&D department and who now has to reconvert himself into a commercial service provider, as in the case of Open Cascade.

At any rate, a change of values is indispensable if success is to be achieved and implies a radical transformation of the professional identities of all concerned. It will certainly be a long and difficult road[10].

[10] React with a presentation within the framework of second workshop, of representative of Alcatel and of Thomson have confi rm

7.3.5 More social than private benefi ts, at least in the short run.

The transformation of a software package into free software can be interpreted as being a more or less significant social benefit according to success of the freed software. However, this could imply a loss of earnings (i.e. possible absence of royalties for the software) and especially investments on the part of the company making this decision. For the company, the profitability of these expenses is not always obvious and can only be determined over the long term. Moreover, it appears rather clearly that, in the two cases studied here, the projects can move ahead only because the entities which undertake these projects are supported by powerful organizations (E.D.F., EADS) for which the additional expenses or the initial losses represent only a small fraction of their sales turnover. As the General Directorate of Industry, Information Technology, the Post Office and the Ministry for the Economy put it, "an open source strategy, requiring a large investment, without receipts coming directly from the developed product, is obviously not a risk-free operation for its promoter. At the same time, it can have significant repercussions for the community of users." ("A Key for the Future: Six Exemplary cases", *R&D Industrielle*, p. 79).

A situation where the social output of an initiative exceeds its private output raises the question of at least some public financing for this type of activity, especially if it is judged that this type of initiative should not remain marginal. Under such conditions, public financing can be an immense help in beginning a project, in particular when the project does not require the freeing of the source code of an existing software but rather the creation, from scratch, of a free software package. A good example of this is the case of some German dentists who wanted to develop, a free software-based machining device for manufacturing dental prostheses. The idea was that, through a system of digital sensors, dentists could make prostheses directly in their surgeries. The project cost was five million marks and was to be financed by a subscription of dentists. It failed because they lacked the last 300,000 marks.

Conversely, the Salome project ("Digital simulation with Open Source Architecture Software and Evolution Methodology") represents "a very interesting and undoubtedly rare case of implementation of a true industrial strategy based on free software" ("A Key for the Future: Six Exemplary Cases", *R&D Industriel*, a report presented by the General Directorate of Industry, Information Technologies and the Post Office, p. 60). This project is located at the interface between CAD and digital simulation, sectors where the two cases studied above occurred. This confirms the contagion hypothesis whereby free software gradually spreads in those fields where it already exists. The idea that has been adopted in the SALOME project is to build a generic platform of CAD-calculation connections for digital simulation. This platform should comprise several modular and configurable components which are adaptable to all kinds of technological fields. The project is based on one observation: the complexity of modern technological products and the reliability required of them makes increasingly fine simulations during the development phase an absolute necessity. Certain CAD software programs propose simulation modules, but they are generally not sufficient and do not cover all fields. The majority of industrialists thus use specialized calculation software, or solvers, which are either developed for them or bought on the market. But linking the CAD and these simulation tools is anything but easy. The information introduced into the solvers and must be done in a very standardized manner which explains why the "data preparation" must often be done "manually", a phase, which serves to establish the link between colossally powerful data-processing tools, can, in certain cases, take up to as much as 50 % of the study.

The SALOME project is master-minded by the RNTL (National Network of Software Technologies under the auspices of the Ministries for Industry and Research) which finances up to 40 % of its activities. The developments carried out within the framework of this project are in the form of free software with a

---

the diffi culty cultural that represent for their developer to publish their source code source, to it make divide outside and inside companies. It is, for them, the principal barrier to the setting into free of some their software.

license that is based on three principles: a) access to the sources of SALOME components is free-of-cost, b) any correction of anomalies carried out in SALOME must obligatorily be published in "open source", c) all those who use SALOME are free to select the user license relative to complementary components that they has developed. The choice of a free diffusion model, with technical support and free access to the source code, is designed to create a community of users. The idea is for the latter to become prospective customers of the associated services. This is part of RNTL's overall strategy to promote free software "in fields where French industry is not well represented [and where] free software provides an effective means to expand technological innovation and to increase their competitive edge." At first sight, it might seem difficult to find partners who would be ready to spend significant sums to develop software intended to be distributed for free. However, in the SALOME project, this idea was enthusiastically shared by nine partners, all of which are eminent representatives of French industry (Open CASCADE, Véritas Bureau, Principia, Cedrat, E.D.F. and the Joint Research Center of EADS) and public research (ECA, Institut National Polytechnique of Grenoble and the Pierre and Marie Curie University, Paris VI). The software platform must be available by September 2002 and there is a convention to continue developments, in the form of free software, after 2002.

The passionate enthusiasm generated by this project is linked to its difficulty. Indeed, the software tools that have been developed must be able to accept information which is specific to a range of technological fields like mechanics, hydraulics, electromagnetism, etc. These fields are so numerous and so specific that the R&D work necessary to development such tools might appear to be beyond the reach of any single editor. This explains why the position sought by SALOME was not already occupied by any competitors, whereas the need is obvious. The success of such a project presupposes that it will become a de facto standard and thus gain quick acceptance so as to impose a technology which will become an essential reference for a majority of users. Otherwise, if the standard project is supplanted by a competitor, the work carried out to develop it will have been a total waste of time. Moreover, the generic aspect of the SALOME platform is crucial in reducing training time for the use of the tools that have been implemented. For the users, the possibility of studying phenomena that have been more efficiently linked will contribute not only to the reduction of costs but also to the duration of the studies. At the same time, it will increase the quality of simulations by promoting interoperability between the simulation codes.

In addition to the importance of public support for this project, a significant factor of success is the fact that it has additional advantages for various participants. Consequently, it corresponds to a combination of both models described above. For the industrialists, the objective is to make their investments more profitable in order to develop digital simulation of calculation codes in the various fields which are at the heart of their commercial activities. This is made possible thanks to the possession of an integration tool in a global platform with CAD software. Moreover, once a decision on a calculation code has been made, it is very difficult to go back on it because the simulations often depend on calculation codes that have been tested repeatedly before. For these future users of platform software, the participation of their teams in a joint development effort offers a chance to monitor technological development by directly contributing to making tools that are necessary within the framework of R&D in their sector of industry, whether it be in the fields of oil, aeronautics or electrical engineering. On the other hand, for Open Cascade, SALOME represents a natural extension of its software platform and it hopes to make its participation in the project profitable through the sale of dependent services. An example of this is the ECA which has already concluded contracts with Open Cascade for the integration of SALOME.

7.4 Conclusion.

The analysis of the software economy has been fundamentally marked by *the permanent coexistence of several "production worlds". In turn, these worlds depend on different conventions that are centered on a*

specific type of product and offer subtle answers to the critical questions concerning software economics. This makes it possible to highlight the reasons for the diversity observed in this domain. The study of the dynamic evolution currently under way shows that the existence of several *production worlds* can indeed be a durable phenomenon. In particular, the originality of free software production and its articulation with commercial forms of production constitute a positive phenomenon for the entire software economy. This situation allows for a certain amount of diversity of software products and the advent of a true "customized approach" towards the production of software on a massive scale.

However, it is not clear whether the *ex nihilo* creation of software, which is free from the very beginning -like the majority of current free software packages- is sufficient to respond to the importance and the diversity of the needs. From this point of view, there is another solution which could guarantee the existence of free software, namely the transformation of private software into free software. This could be an interesting complementary solution. Some pioneering companies have taken this route recently. The new economic model towards which they are evolving is undeniably coherent, but it is obviously too early judge the efficacy of such trends since the examples are all too few to draw any conclusions concerning the practical success of such strategies. In any case, it appears to be clear that such evolution takes time and requires significant resources before possible positive results can be observed. It will be necessary to follow these cases attentively because their success or failure will have a considerable impact on the extension of these strategies to other interested companies which want to double-check the credibility of these new models before adopting them. In the meantime, in order for free software to spread to the majority of the software economic fields - in particular those sectors dominated by American companies and/or the sectors where "private" software is not responding satisfactorily to needs (as in the position filled by Salome) - the State will need to continue providing assistance.

Licences: strategic tools for software publishers?

LAURE MUSELLI,
University of Paris XIII, laure_muselli@yahoo.fr.

With the rise of the free software movement and as a consequence of the burgeoning number of information processing licenses used by software editing firms, one may wonder what drives a firm to choose one type of license over another. Given the characteristics of the software industry, connected to a network industry in which battles between standards rage, one might ask oneself whether licenses might not be a powerful strategic tool which, if subtly used, could help companies meet their objectives. Consequently, one of the main features of the software industry resides in the fact that a company seeking to profit from a software program it has conceived cannot function according to traditional commercial logic. Indeed, it does not sell its software as if it were a product, but makes from the licensing agreements with any individual wishing to use them. In this way the software is assimilated to a technology. Rather than selling the software, the company simply concedes the right to use its technology. It is the computer license, a contract granting the user the right to use the software, which defines the conditions under which its use is possible. As such, there are accompanying obligations that are more or less restrictive. For example, some companies promote a strategy whereby they exploit copyrights to the utmost by imposing high fees and strict conditions of use. Others bet on a panoply of strategic aspects, such as standardization for example.

This objective of this article is to show that there is a direct connection between certain types of clauses and precise strategic objectives. Needless to say, one must not forget that the static character of a license is in contradiction with the evolutionary nature of these strategic objectives.

In the first part, we will review the characteristics of network industries and we will present the strategies that are generally implemented to exploit these characteristics. After having drawn a picture of the many existing software licenses, we will establish a typology of strategies that are likely to correspond to these licenses. In the third part of this article, we will finally propose a methodology allowing the reader to link these various strategies to the license clauses. In so doing, we will apply concrete cases to software editors such as Troll Tech or Sun Microsystems.

8.1 Characteristics of the software industry: increasing output and standardization.

In order to answer the questions which we will tackle in this article, it is necessary to understand that a software is generally composed of several modules which make a product "improvable", modifiable and evolutionary. This leads us to classify the various types of software into two categories.

The first category is what we will call "software technologies". This is a vast category in which one finds the operating systems, computer languages and libraries. The software classified in here is intended to be used to develop other software products (as in the case of computer languages or libraries) or to make them function (this is the case for operating systems which are essential in making some kinds of application so that the software can function on a computer). As such, on analogy with industrial technologies, this makes them closer to technological software. Just as industrial technology has only one goal - the manufacture of products or the management of production, similarly, software technologies are only geared to the management or the production of derived software products.

The second category includes application software, programs that are used as end products by the users and are derived from software technologies and other software from which they are conceived. Some examples of these are office automation software such as word processing, calculation software, multi-media creation or page-setting computation software. This type of software can be compared to a traditional product resulting from an industrial technology.

In this article, we will focus on the first category —software technologies— because of their evolutionary and modifiable character. This is a characteristic that cannot be attributed to software products. As a consequence of the nature of software technologies, they are important to editing firms which see in them a means

of controlling derived markets —if they succeed in imposing their own standard. One thus understands better their interest in deploying aggressive strategies which are aimed at benefiting from the natural mechanisms which exist within network industries— in which the software industry itself is a part.

8.1.1 Characteristics of a network industry.

If the strategic concerns of the editing firms which are working on this type of software appear to be so significant, it is because they are closely linked to the characteristics of this sector, which are, in turn, similar to a network industry. On the one hand, there is the presence of powerful network outsources which stimulate feedback and, on the other hand, have a significant possibility of locking in users.

Strong network outsources inducing feedback.

As Marie Coris has already stated in this book, Information Technology is an industry in which the value of a software program is very often dependent on the number of people who are already using it (this is what is called the user network). Thus, the entry of a new "member" in the network will benefit everyone by creating a network outsource. Assuming they are of equal value, a newcomer who must choose between two networks will tend to enter the one with the most people. On the software technologies market, where the network consists of users of the same technology (languages, operating systems, libraries, etc.), this principle is largely proven insofar as the user of an isolated technology can neither communicate with other members, nor can he/she propose interoperable derived products. This produces a "feedback" effect : the more a technology is appreciated and adopted by people, the more others will anticipate that it will become the standard. They will thus adopt it to benefit from the network outsources mentioned above. A virtuous circle is then formed and technology imposes itself as a market standard. Inversely, however, when a technology is declining, its fall will be accelerated. Indeed, if the first purchasers make only moderate use of a technology, the other potential users will anticipate that it will not become the standard and may even disappear. Consequently, they may choose not to adopt it, because of the absence of network outsources. This time, it is a vicious circle which is formed, involving the loss of technology in the battle for a standard.

Thus, the feedback effect is a consequence of the desire by the purchasers to benefit from the network outsources or, put another way, to belong to the broadest network and to acquire the technology which will end up being essential as de facto standard. The success of a technology often hinges on the reaction of the first critical mass of users who determine whether it survives or disappears.

Possibilities of locking.

The second characteristic of network industries and particularly of the information technology industry is locking in the users in situations where the costs of change are too significant, factors which oblige them to retain adopted technologies. This locking in can concern *permanent equipment, which provokes increasing changeover costs over time, in particular because of the compatible complementary equipment acquired over the years* (this is especially the case for Microsoft, which purposely discourages compatibility between application software, other than its own with its Windows operating system). Locking in also concerns *training, which, in the event of change generates training costs or losses of productivity* (like the fact of learning how to use Linux in stead of the Windows environment). Finally, locking in concerns *information and data bases, for which a change of equipment provokes conversion costs of data to new formats* (one can quote the example of specific file formats common to each kind of software) Shapiro and Varian [2000]. These characteristics support the establishment of standards guaranteeing that the firm which succeeds in establishing them will dominate the market. One then understands better the efforts of software edition market firms to establish strategies aimed at imposing their technologies at all costs to all users. The only alternative is to be dominated

by their competitors. Considering how vital it is for a firm to establish its software technology as a standard in order to dominate the market, the firms will seek to benefit from user feedback as a negotiable instrument by developing it quickly and then by amplifying it. It is in this manner that they hope to establish strategies which will aim at the installation of de facto standards.

8.1.2 Strategies benefiting from feedback effect.

Shapiro and Varian [2000] highlighted two strategies making it possible to benefit, in different ways, from the feedback effect: control and openness.

The control strategy.

For the firm, this means being in complete control of the diffusion and the evolution of its technology. This usually results in monitoring the improvement of the technology as well as the rate at which superior versions of it (for example, Microsoft) can be made available. By controlling the diffusion of the technology, the firm is certain to be the only player to deliver a user license. The final point is the control of the compatibility of the firm's technology with that of other firms. In so doing it can prevent access to the source code or voluntarily introduce some incompatibility with the competitor's system.

Once the technology has been adopted as a standard, the advantages of this strategy become self-evident: the firm does not have any concessions to make in terms of compatibility, because it controls all the cards. Its position being thus reinforced, serious competitors have little chance to resist or threaten it. In addition, once the customers are locked in, the firm is able to fix the prices and the conditions of its licenses as it sees fit. Finally, it is not obliged to constantly improve its products, because the customers are locked in and there is an absence of serious competitors on the market.

But one must admit that the installed base is difficult to realize. Indeed, the prospective customers should be wary of this control strategy which is likely to put them in a situation of total dependence which could lead them to choose a competing technology. Critical mass is difficult to achieve under such conditions.

The openness strategy.

The strategy of openness is the opposite of the control strategy insofar as it encourages offering more favorable conditions to the users. This strategy can itself be broken down into two sub-strategies which are standardization and total openness.

Normalization.

Normalization consists in entrusting the choice of a common standard to an standardization organization such as the ISO or the ECMA (European Computer Manufacturer Association) whose aim is to bring all the competitors present in the committee to a consensus. It occasionally happens that the choice of a technology is already made by the industrialists before the organization actually intervenes. In such cases, the organization guarantees the future openness of the standard. In other words, it is this organization which dictates the future evolution of the technology.

A firm which imposes its technology as a "de jure" standard (norm) will be able to reach an installed base relatively quickly by avoiding a costly - and ambiguous - battle over standards since the users will have a guarantee regarding the durability and compatibility of this technology. However, this type of strategy results in the loss of control of the company in favor of the standardization organization to which the choices of the future evolution will be entrusted. Thus, the profits gained as a result of the firm's monopoly will be less than would be the case if a control strategy were adopted insofar as all the other firms in this market are also able to

use the standard technology. This competitive environment makes it impossible to "lock-in" users concerning an offer and, consequently, to lower prices. In addition, the firm's R&D costs are likely to rise in order to produce quality products in order to preserve its market share.

Total openness.

Total openness is the situation where a firm decides to place its technology at the disposal of all users, without any restrictions, so that other firms can manufacture compatible products. This strategy is founded on the principle of **a total absence of "lock-in".** In so doing, users acquire a sufficient installed base in order to quickly benefit from significant feedback effect from the users. The latter, who have by now been won over by the absence of "lock-in", will adopt technology more readily which allows the firm to acquire an important installed base.

The disadvantage lies in the fact that, even though its technology has been adopted as a standard, the firm really no longer has an advantage over its competitors which can manufacture compatible products just as well as it can. However, when the outcome of the battle over standards seems dubious, the attraction of a completely open product then remains one of the only real chances for the firm to impose its standard, even if, through the total loss of control of its technology, the firm cannot make any profits off the technology itself. To benefit from this strategy, it thus must have the competence and the technological advance necessary to offer complementary products or paid services and to be remunerated in these ways.

Once the strategic choice is made between control and openness, the firm will have a great number of tools to implement its strategy regarding the imposition of a standard. Later in this article, we will study one of these, the information technology license. We shall try to show that it can prove to be a powerful strategic tool to benefit the firm.

8.2 License strategies: a typology.

Before concentrating more specifically on the various existing licenses, we will start by pointing out some concepts essential to comprehension of typology presented here.

8.2.1 A few concepts.

The intellectual protection system to which the software is submitted is the copyright. In other words, any company that has designed a software program benefits from intellectual property rights guaranteed by the copyright. Licenses are nothing other than the concrete realization of the use the firm intends to make of its intellectual property rights with respect to its software[1].

Thus, the firm wishing to profit from the use of a software program it has designed is not functioning in a traditional commercial framework. Indeed, it does not sell its software as if it were a product, but it makes licensing agreements with any individual wishing to use the software by assimilating it to a technology: rather than making a sales contract, the firm makes a contract conceding the right to use its technology. That gives the user the right to use the software. However, it also imposes more or less restrictive obligations as well as relatively high fees to pay.

Some firms choose a strategy which allows them to valorize copyrights to the fullest extent by imposing high fees as well as strict conditions of use, whereas others make other strategic choices such as standardization, for example.

[1] It is a question of studying the strategic consequences of the license clauses, here. Mélanie Clement-Fountain makes their legal analysis in the following part. Thus, in this article, the terms "copyright" and "droit d'auteur" are regarded as synonyms. On that topic, considering economics point of view, we will cite Gallini and Winter [1985], Katz and Shapiro [1986], Bousquet and Wolkowicz [1994], Kotabe et al. [1996].

For example, the Copyleft concept created by Richard Stallman, the founder of the Free Software Foundation in 1980, plans to offer a protection tool to the GNU project in order to stop software which has been "copylefted" from falling into the hands of firms which might attach proprietary and restrictive operating licenses to these programs[2]. If the copyright associated with a property license often has the role of restricting the freedoms of the software user, it is, on the other hand, the clauses of the copyleft license which provide additional freedoms to the copyright. These supplementary clauses thus offer freedoms to a "copyrighted" software, whereas those of a proprietary license, which is a supplement to the copyright, authorizes monopoly income. It is thus not the copyright, a right given automatically to any software author, which determines the monopoly income or the free character supplementing the software, but rather the license whose clauses limit freedoms or, inversely, preserve them.

This copyleft principle will be used with different clauses according to the types of licenses which we will review.

8.2.2 Numerous licenses.

"Public domain" licenses.

The products under control of this type of license are not free from every intellectual property law because they are protected by a copyright. However, the added clauses grant (to the users) unlimited rights to **use** it, **copy** it, **modify** it, **amalgamate** it [with other software], **publish** it, **distribute** it, **sub-license it** and/or **sell copies** of it and to allow people to whom it is distributed to do likewise.

Free licenses.

Two associations, the FSF and the OSI, drew up a list of criteria making it possible to define the concept of freedom and to determine the membership or not to a license in this category. Even if the criteria proposed by these two associations defer to some extent, it appears that their conclusions tend to agree. For the FSF, the significance of "free" can be summarized in 3 points: a) the freedom to use the software (i.e. to execute the program), b) the freedom to have access to the source code, whatever the objective might be (i.e. to study the program functions, adapt the software for personal needs, improve the program by correcting any errors), c) the freedom to distribute copies of the software, by whatever means, **whether free of charge or not**. Thus, all software governed by a license excluding even one of these freedoms cannot be qualified as a "free" license. The word "freedom" implies here that it is by no means necessary to ask for the authorization to carry out any of these actions: the user automatically subscribes to the license right from the moment he/she carries out one of these actions. In addition, a user cannot, under any circumstances, place a derived product of this type of software under a different license: all derived products must remain free.

[2] In practice, putting software under copyleft is done in 2 stages. First of all, it has to be placed under traditional copyright. Then, it has to be coupled with a license, that is to say, additional clauses relating to the software distribution conditions. These conditions, which can vary according to the license that is used, generally stipulate that any user has the permission to carry out, to modify or freely distribute the software, insofar as he/she transmits these same rights to the users of all the copies or derived work of the aforesaid software, without adding any restrictions at all. This means that any user who has modified the source code of a software under copyleft must place this new software under a free license. In order to protect free software, the use of the copyright is the best solution that has been found to date. Indeed, if the author decides to give up his copyright to allow the diffusion of the software, the program falls into the public domain. This authorizes any individual or company to diffuse the software, on the one hand, but also to, eventually, have the right to transform it into proprietary software, and thus restrict its use.

The proprietary licenses.

This type of license is classically used by a majority of software editors. The characteristics of this type of license are as follows: fee is required for its use and prohibition to copy the software, access the code-source, modify the software and to distribute it to anyone. Contrary to free licenses, the copyright includes restrictive clauses limiting the diffusion and the modification of the software.

Hybrid licenses.

This type of license simultaneously combines the copyright with clauses borrowed from the free licenses and other clauses borrowed from proprietary software. Consequently, this type of license can be neither completely assimilated with free licenses, because it does not meet all the conditions stated by the certified organizations, nor with the proprietary licenses.

As an example, one can consider Sun Microsystems' SCSL license which is applied to the Java programming language. This license comprises clauses which are compatible with free licenses such as those concerning the right to use and copy software programs freely as well as have free access to the source code. On the other hand, the clauses concerning distribution do not satisfy the freedom condition: the distribution by third parties of products derived from those protected by the SCSL remain free if they are made free, but this is no longer the case from the moment a person wishes to profit financially from this activity. In this case, Sun Microsystems imposes the payment of royalties/fees, but also obtains the power of certification of derived products without which the firm does not have the right to distribute its products.

The reasons firms choose this type of license remain ambiguous. They generally do hesitate to communicate on the free character of their products, which are nonetheless subjected to hybrid licenses. When questioned, they especially criticize the viral effects of the LPG.

It appears that there are numerous types of licenses and each category has even a greater number of different licenses. Certain firms do not hesitate to create their own. Faced with such a multitude of licenses, one wonders what the criteria are which can lead firms to choose public domain licenses (free, hybrid or proprietary), if the choice of each clause is pointless, or if on the contrary, it is carried out with a precise strategic purpose. To answer this question, it is interesting to consider the strategic lessons learned from the study of the technology license agreements.

8.2.3 License strategies: a proposal for a typology.

In order to show that licenses are conceived or chosen with precise strategic purposes in mind, it first appears necessary to define which types of strategies are likely to be successful through the use of licenses. In order to accomplish this objective, it seems relevant to refer to Bessy and Brousseau [1998]'s article in which they attempt to demonstrate that technology license agreements can be drawn up even when strategic objectives are extremely different one another[3].

These authors highlight four explanatory factors of the types of strategies to which the various types of existing technology license agreements are dependent: **policies of the firms regarding intellectual property**,

[3]It could be stressed that IT companies do not use technology license agreements, but rather IT licenses for their products. There are signifi cant differences between the two. Firstly, technology license agreements are made concerning one or more patents which are held by the company. With regard to software programs, the license relates to the copyright, since this is the legal way of protecting software. Secondly, technology license agreements are concluded only between companies. One company is the owner of a patent, another company wants to use that technology. As for software licenses, the licensee can be a company which needs the technology to develop other IT products. However, it could also be a fi nal user, either a company or private individual, that uses the software as a fi nished product: this never is the case with technology license agreements.
However, the similarity that exists between the two, mainly the identical principle which consists in transferring operating rights under certain conditions, allows us to make this analogy.

the dynamics of the process of innovation, **the degree of mutual dependence of the members of the industry** and **alternative organizational forms with respect to the license agreement**. And the characteristics of these license agreements will vary according to the strategy adopted by the firm.

While inspiring ourselves from this work, in the case of the software, we end up with four explanatory factors which we will call patrimonial valorization, control of the competing firms, creation of co-operation (possible source of savings in R&D), establishment of standards.

Patrimonial valorization.

The first type of strategy is undoubtedly the most immediately perceptible one and highlights the patrimonial valorization of knowledge of the firm through its title of intellectual property. The firm which holds the intellectual property rights for a technology benefits from a monopoly income because of its exclusive right to exploit this technology. The royalty/fee received within the framework of a license agreement is thus a form of compensation for losses of part of the monopoly income when the firm yields the right to exploit technology to the licensee. For the licensor, the patrimonial valorization strategy consists in generating income thanks to the direct use of its intellectual property rights, while making license agreements with other firms which are eager to use the technologies produced. The royalties thus resulting from the license agreements can then be used specifically for the financing of new R&D activities. In this case, the proprietary rights represent only assets which make it possible to generate an income.

This type of policy is obviously very widespread in the software edition sector. Indeed, the licenses known as "proprietary" licenses which are associated with software marketed by a considerable number of editors are used with only the purpose of valorization in mind. The conditions of the information technology license agreement vary according to the importance of the power the editor has over the market: the more the matter approaches a monopoly type of situation, the more the license will be able to impose restrictive conditions of software use and impose high royalties. Thus, for firms which benefit from a quasi-monopoly of some of their products, as in the case of Microsoft, the license clauses can be very restrictive and very high fees can be requested because no equivalent competing product (not only from the point of view of quality but also compatibility) is present on the market.

Controlling the competition.

For the firm, the second prospect consists in considering the license agreements as a *means of positioning the firm on the market and controlling the competition*. This is accomplished in several ways: a) by making the licensee pay significant royalties in order to control costs, b) by imposing grant-back clauses on the licensee (i.e. the *obligation to reassign exploitation rights to the licensor for the innovations that have been derived from the initial technology*) in order to control its technological progress. It can also be accomplished by fixing conditions allowing it to dissuade threatening firms from entering the market or by excluding some the rival firms from the competition.

This type of strategy, which we will henceforth call "strategies controlling the competition", is applied in the case of IT proprietary licenses, in particular through the use of certain restrictive clauses. One can mention, among other things, those clauses prohibiting reverse-engineering. The latter make it possible for the editor to control the technological progress of its competitors by not giving them the possibility of creating derived or compatible products and by reserving the exclusivity of this possibility to him/herself.

The *grant-back* clause is found in certain licenses which we will describe as hybrids and which combine characteristics of a free license and a proprietary license. Thus, for its Java language, Sun Microsystems wants - for certain strategic reasons - to grant a free license to Java users if products which are derived from it are intended for personal use or for free distribution. On the other hand, it imposes a particular clause requiring the payment of royalties if the licensee seeks to benefit from the sale of products derived from Java.

Thus, technological progress is encouraged by Sun Microsystems, but the moment the licensee tries to profit financially from this technological progress, the conditions of the license change.

Creation of co-operation, sources of possible savings in R&D.

Given the costly, expensive, rapid and dubious character of innovations, it might be in the interest of certain firms to develop a co-operation strategy. In the case of a process of incremental and cumulative innovation, the production of the innovation provokes a feedback effect. The search for a leadership position encourages the adoption of either integration strategies or cooperation agreements within the framework of hierarchical relations. In the latter case, it is a co-production contract which is drawn up, the consequence being that the firm provides the technology to the licensee: the latter takes on the role of a subcontractor charged with producing complementary assets. The licensor benefits from the experience of the licensee and preserves his/her advantage in terms of innovations by imposing grant-back clauses and the supply of assistance and counseling services. This allows the R&D of the licensor to profit from the feedback effect of the licensee's production activities.

Considering software production costs, one can understand why a firm can decide to encourage a community of programmers to work on one of its software programs in order to enrich and debug it. In such a case, the firm will use a license whose clauses will provide powerful incentives for the programmers to cooperate and encourage diffusion; these clauses can concern free use, free modification, free distribution, as well as access to the source code, just as in the GPL. In this case, the firm places itself in the position of a licensor of a technology: the software or the language. The "community" of programmers plays the role of the licensees, and can be seen as subcontractors in a co-production contract. This allows them to produce complementary assets such as derived products or improved or debugged versions. In this case, the firm profits from the feedback effect insofar as it benefits from these improvements of its own product without requiring any investments in R&D.

Establishing a standard.

In the cases where powerful network outsources between technologies or technology users characterize a market, we pointed out it that mechanisms for increasing adoption returns can exist. Certain strategies will thus focus on these increasing adoption returns in order to acquire a leadership position by making their technology a practical standard, in a sector which plays according to the "winner takes all" rule. The reasoning behind this is simple: the more a technology is diffused, the more it is adopted by a large number of users. It becomes more attractive to users who see in this development a guarantee of durability and stability. The openness strategy, which we presented previously, makes it possible to win a war of standards.

In information technology industry, firms try, by various means, to get their software, their language or their information technology accepted by the greatest number of users, in order to impose them as standards to the detriment of their competitors. In order to accomplish this, they can choose an open, attractive license policy, (i.e. to concede licenses for a maximum number of users), under conditions which will entice them to choose their technology rather than that of the competitors. In the case of the software or languages, these conditions are translated by clauses which offer low prices or even make cost-free proposals. They also provide access to the source code and the freedom to modify it. The firms choosing this type of strategy generally select either proprietary licenses, free licenses or hybrid licenses. The examples are numerous: Sun Microsystems for Java, Tech Troll for Qt, etc... As in the case of traditional technologies, this type of strategy is often used by the software editors to make joint profits, most generally by providing services, maintenance, complementary software subjected to proprietary license or by creating a system of multiple licenses according to the type of activity of the user (Tech Troll with Qt for example).

8.3 Towards a license analysis methodology.

It appears obvious that a firm's strategy is seldom connected with the one of the four "pure" cases that we have just described, but often consists of a mix of strategic components borrowed from these four types.

If we consider that each license clause corresponds to a precise strategic contribution to the firm, one can consider the information technology license to be a grouping of these specific clauses and, by extension, a grouping of strategic components borrowed from the four different types of strategy. It thus appears possible to decipher the overall strategy of a firm by analyzing the various clauses present in its product licenses and by coupling them with the strategic components which they serve.

This observation leads directly to a beginning of strategic analysis methodology of licenses which we will base on the four strategic components described above. As a principle it associates each clause of a license with one or more strategic components which it serves. We will apply this methodology to two concrete examples: the case of the Qt of the Troll Tech library and that of the Java language of Sun Microsystems. To simplify things, we will symbolize the four strategic components hereafter in the following way: S1 for patrimonial valorization, S2 for the control of the competing firms, S3 for the creation of co-operation agreements and S4 for the establishment of standards.

8.3.1 The case of the library Qt by Troll Tech.

Qt is a library perfected by Troll Tech. It is used by KDE for its "desktop environment"[4]. Project KDE plays the role of producing a free desktop environment, placed under GPL, in order to provide free software with a free environment which is convivial as the environment enjoyed by proprietary environments. However, Qt, the library used by KDE, was not placed under a free license by Troll Tech. This went against "the spirit" of the free community and the most virulent members roundly criticized this development. Since then, the Qt license has evolved: we will thus divide the history of Qt into three phases, and we will apply our methodology to each of these three phases.

Phase n°1.

First of all, it should be recalled that Troll Tech applies a multiple license policy for Qt. It justifies this by the fact that it wants to have, on one hand, a free policy for its products and, on the other hand, a proprietary marketing policy in order to profit from the economic fallout which could be generated by other firms using Qt's proprietary derived products. Thus, it has created for such occasions, a license for noncommercial use called the "Troll Tech Free Software License" and a proprietary license for all the firms which create proprietary products derived from Qt (without access to the source code). Qt's Troll Tech FSL version is thus directly intended " to be mixed " with free products, whereas the proprietary version "will be mixed" with non free software.

If the name of Troll Tech FSL itself indicated that it conferred a free character on Qt, in reality, without it being a proprietary license, it could not be considered to be a free license either: if the free use was respected, on the other hand, modifications were prohibited and free diffusion was limited to derived products whose source code was accessible.

If one takes again the clauses which are characteristic of this combination of licenses and if one associates them to the strategic components which they serve, one obtains the following result:

[4] A "desktop environment" is used to configure the graphic aspects of a screen and is composed of a window manager (making it possible to configure the frame and window management) as well as tools such as bars or drop-down menus, thus improving user-friendliness.

➢	Free use	➜	S4	
➢	Prohibited modification	➜	S2	(goes against S3)
➢	Availability of the source code	➜	S4	
➢	Authorized distribution	➜	S4	
			+	
➢	Proprietary license	➜	S1 + S4	

It appears clearly that the establishment of Qt as a standard remains the principal objective of Tech Troll, considering the strong presence of clauses supporting the S4 strategic component: a) free use insofar as it encourages the adoption of the software, access to the source code, which also goes in this direction, since it guarantees a certain transparency of the product to the user, and finally the authorized distribution encouraging the diffusion of the product. This coupling with a proprietary license also encourages the diffusion of the product to a maximum number of users since it makes it possible for Troll Tech to limit itself to the free community. It also allows it to work with users in firms who want to produce the non free software based on Qt. Since Qt and Troll Tech were not very well known at the time, one can easily understand why the top priorities were diffusion and democratization in order to make this library a standard. In addition, the control of the competing firms seems to have been, at the time of phase n°1, one of the principal concerns of Tech Troll. Indeed, the clause prohibiting the modification of software refers directly back to S2, since it removes any possibility for an individual or a firm to "clone" the library. If one puts oneself in the shoes of Tech Troll at the time, this decision is totally logical: being a small firm offering a little known product, it could not run the risk of seeing a more powerful firm clone its product and impose it as standard through its superior financial means and reputation. Lastly, the final component supported by Troll Tech is S1, the patrimonial valorization of intellectual property rights over Qt. The existence of the proprietary license designed for users seeking to produce non free software that is derived from Qt technology, enables the firm to make a profit off the royalties. It is significant to note the absence of the strategic component S3 (i.e. the creation of co-operation agreements) which is not represented in any of the clauses of the Troll Tech FSL. This can be explained by the priority that was granted, in particular, to the strategies employed to establish a standard (S4) and the control of the competition (S2). This was necessary in a launching phase, sometimes to the detriment of the "co-operation" component (S3). The clause relevant to the modification prohibition provides proof of this since it makes it possible to control competitors, but at a price which prohibits the users from making any profit (S3).

Phase n°2.

KDE was supposed to promote free software programs. The "free" community rose quickly against the fact that the Qt library was a non-free product. Very quickly, this community exerted pressures on Troll Tech so that it would replace its Troll Tech FSL license with the GPL license. The argument was that the prohibition of modification of the library was a serious handicap for the development of powerful software. One of these pressures was the launching of the GNOME project, intended to develop a completely free desktop environment based on a library called GTK, equivalent to Qt, but under GPL license. Tech Troll did not yield and modified the license, thus not protecting Qt under GPL but under QPL (Qt Public License), still a non-free type of license but which authorizes modifications in the form of "patches". In addition, it added a clause stipulating that these modifications could be re-used by Troll Tech in this same product or other products, under different licenses.

Here is, for the new combination of licenses, the representation of each specific clause associated (with) its strategic component:

➢	Free use	➜	S4
➢	Authorized modifications if patches	➜	S2 + S3 + S4
➢	Source-code availability	➜	S4
➢	Authorized distribution	➜	S4
➢	Modifications usable by Troll Tech	➜	S3
		+	
➢	Proprietary license	➜	S1 + S4

With this new combination of licenses, the S4 component is privileged because it retains the clauses guaranteeing free use, availability of the source code, authorized distribution and the proprietary license. It is still reinforced by authorization of modifications in the form of "patches", insofar as this clause constitutes an incentive with the adoption of Qt by users who would have, in the preceding phase, been put off by the total impossibility to modify the library. The S2 component is also present through this clause, since, thanks to this solution, the users' desire to be able to modify the library does not affect the control which Troll Tech can exert on its competitors: cloning is still impossible. The S1 component is retained thanks to the proprietary license. Finally, the authorization to modify the software also results from the appearance of S3 components in the license: the co-operation of users to enrich Qt, as well as other possible Tech Troll products, is a strategic approach that, this time around, has not been neglected by the firm.

Phase n°3.

In 2001, Troll Tech finally decided to place Qt under **GPL** while retaining the principle of coupling with a proprietary license. Here are the consequences of this change of license in strategic terms:

➢	Free use	➜	S4
➢	Free modification	➜	S4 + S3
➢	Availability of the source code	➜	S4
➢	Authorized distribution	➜	S4
		+	
➢	Proprietary license	➜	S1

The S4 component continues to be largely represented by the free use, the availability of the source code as well as the authorized distribution. Moreover, contrary to the preceding license, it gains more authority through the fact that the freedom to modify the software is conferred by the GLP license.

The S1 component is also present through the royalties paid within the framework of the proprietary license.

But Troll Tech reinforces the S3 component of its strategy because, if it removed the clause authorizing it to freely make modifications to Qt, it benefits, nevertheless, through the free modification clause, from the enrichments carried out by the Qt users. One can see in this strategic evolution the Tech Troll desire to see its library enriched by a community of programmers since it is faced with potential competition by GTK (GNOME): **Troll Tech may have been frightened by KDE desktop environment competitor and decided to succumb to the request of the free community in order to avoid losing market shares to GNOME.** This does not mean that **if KDE had won over a large number of users**, the latter **would have** been **attracted to GNOME that easily,** whether **Qt** had been **under GPL license or not**. Indeed, Troll Tech may have feared that its library would become inferior to that of its competitor whose own library was constantly being improved by a community of programmers. To encourage the community to improve Qt while placing it under GPL could be a good way to avoid being technically surpassed.

Lastly, one notes the disappearance of the S2 component in Tech Troll's license strategy which, by adopting the GPL, gave up the control of its competitors by ceding the modification prohibition. Indeed, even if

QT was once a little known and financially insecure company, today it is a quasi-standard and Troll Tech has acquired sufficient fame and maturity to enable it to yield to the free community by placing Qt under the GPL without fearing the creation of a competing version by a large firm.

In short, from now on, the GPL is more of an advantage than a disadvantage for Tech Troll, insofar as it encourages the use of Qt (in particular by students who do projects under Qt and then impose its proprietary version on their employers). In this way, sales of the commercial license are stimulated by imposing the standard a little more and by enriching the product through the work of the programmers.

8.3.2 The case of the Java language of Sun Microsystems.

The SCSL license was created by Sun Microsystems for Jini, the Java protocol of communication between peripherals, and it was then applied to several other products, in particular Java.

If Sun Microsystems communicates about the free character of the SCSL license, according to the regulations of the Free Software Foundation or OSI, it is not, however, a free software, insofar as the 3 basic criteria are not strictly respected. Indeed, even if the product can freely be used and copied by any user, it is relative to the conditions of distribution that significant restrictions are imposed:

- on the one hand, the distribution by third parties of products derived from those protected by the SCSL is not free: the distribution remains free if it is done free of charge, but it is no longer the case as soon as an agent wishes to profit financially from this activity. In this last case, Sun Microsystems imposes the payment of royalties, but also grants a certification for derived products, without which the firm does not have the right to distribute its product. Thus, Sun Microsystems keeps a tight hold on all the products initially developed from its Java protocol;
- in addition, the following restriction is specified: if the corrections of errors must obligatorily be placed under the SCSL license before being given to the community, other kinds of modifications (such as the improvements) can become Sun owners, even if they would be the fruit of the work of members of the community of developers not working for Sun.

➢ Free use ➜ S4
➢ Free modification ➜ S4 + S3
➢ Cost-free distribution ➜ S4
➢ Royalty ➜ S1
➢ Certification ➜ S2
➢ Usable modifications ➜ S3

Sun Microsystems' SCSL largely supports the S4 component through the clauses of free use, free modification and free distribution if the latter is done free of cost. These three clauses thus translate the desire to diffuse Java as widely as possible, to democratize it and encourage its adoption. One might conclude that this strategic aspect was of great significance in placing Java under this license.

Contrary to the GPL type of free license, this license also makes it possible to combine the S4 component with the S1 component dealing with patrimonial valorization. In order to do this, it discriminates between users and differentiates those users who redistribute the derived products free of charge and those who draw financial benefits from the sale of products derived from Java. For the latter, the redistribution is not free, but requires the payment of royalties to Java, which also imposes a "grant-back" clause (i.e. the payment of royalties for any profits realized with these sales).

According to the certification clause, it is the S2 component, which deals with the control of the competing firms, that Sun Microsystems has set up. Every derived product which is to be sold must be presented for

approval and the choice of whether or not it will obtain Java certification is strictly regulated. In this way, Sun controls the evolution of its products and the use which can be made of them by its competitors and can thus eliminate the projects which it considers detrimental to the strategy of the firm.

Lastly, this license does not neglect the S3 component in the sense that the free modification clause is founded on the co-operation between Java users, thus making it possible to constantly improve it. This co-operation is all the more beneficial to Sun in that the clause, which authorizes it to use these modifications in any product and under any license, allows Java to improve, through the work of the community of programmers, all of its products, including those subjected to proprietary licenses.

Contrary to the Qt example, where the license evolved progressively as Troll Tech was better known, Sun Microsystems, in spite of the pressure of its competitors and of the free software community, was anxious to see a standard controlled by only one firm. To date they have not agreed to place Java under a free license. Indeed, since the system adopted excluded the idea of the double license, even if the product were placed under the GPL, for example, it would not allow Sun to preserve the S1 and S2 components, a decision which would deprive it of some of the financial profits obtained from the sales of derived products.

The study of these two examples clearly shows that it can be in the firm's interest to use a hybrid type of license. The advantages would be to preserve, within the same license, or the system of a double license, interests which would otherwise be incompatible if a pure free license or a pure proprietary license were used. The hybrid licenses are thus born of the desire to benefit from all the advantages obtained from a proprietary license, dependent on the control of technology and patrimonial valorization, as well as all advantages of a free license, which is dependent on the diffusion, on standardization and on the improvement of the product. Hybrid licenses also make it possible to avoid the disadvantages of each of these types of licenses: the absence of direct financial repercussions and the absence of control for the free licenses, and the difficulty of diffusion and adoption as well as the absence of improvement of the product for proprietary software. If one refers to the two strategies proposed by Shapiro and Varian [2000], one sees that the hybrid licenses make it possible for the firm to reconcile both, control strategy and openness strategy.

8.3.3 The strategy and choice of license: a dynamic vision to promulgate.

After having applied this methodology to various licenses, one might be led to imagine that it would be in one's interest to create two types of tools. The first would be a "catalogue" which would associate one or more strategic components with the different types of clauses which have been listed. The purpose of this would be "to translate" any license of a legal language into a "strategic language". This would allow every licensee to determine very quickly the aims of the firm before adopting its software. The second tool would also take the form of a "catalogue", but which would make a list of the various types of clauses likely to serve each identified strategic component. This tool would be used by any entity wishing to find the appropriate license for its product and its strategic aims.

However, this methodology and these tools would give a static image to a license which would be incoherent insofar as the purpose of a strategy is to evolve over the course of time, according to many kinds of factors (the evolution of its share of the market, its fame, the situation in which it places its product - the launching phase or the quasi-standard. It also depends on the number and capacity of its competitors and, finally, the share and pressure of institutional actors such as the Association for the Promotion of Free Software, for example). This vision is too static and can even become dangerous for a firm since it must always anticipate the evolution of its market as well as the use of its software and, therefore, its license if it does not want to find itself blocked in an overly rigid strategy.

Thus, it seems obvious that, when the product is in the launching phase, the strategy of a firm cannot be the same as when the product has reached a phase of maturity. At the time of the launching phase, the first aim of the firm is to succeed in getting a maximum of users to adopt the product, in order to impose it as a standard.

With this intention, the editor could choose a license with relatively non restrictive clauses and impose low royalty fees. On the other hand, when the product reaches a phase of maturity where it is established as a standard, the editor's strategy can no longer be based on acquiring a standardization that has already been acquired, but rather on increasing profitability and gaining control of the competing firms or improving the software by a community of programmers. It is then probable that, depending on the objective that has been set, the clauses of the preceding license will no longer be suitable.

In the same way, the competitive environment of the firm can change. A firm having a quasi-monopoly over its market segment will tend to adopt a valorization strategy and, consequently, a rather restrictive license and with high royalties. From the moment when competitors make their entry on its market, and it sees its positions threatened, it may find it to be in its interest to support a strategy to control competitors or standardization: once again, however, the previously adopted license runs the risk of being inappropriate.

Lastly, as we observed in the case of Troll Tech's Qt library, a strategy can evolve under the pressure of institutions such as associations for the promotion of free software, like the FSF. Some firms can thus be encouraged to re-examine their license and, consequently, even their strategy.

However, just as there are dependencies resulting from technological choices (i.e. an initial choice of a technology has consequences on future choices, rendering the use of certain technologies impossible), in the same way, these dependencies are likely to appear in the licenses: according to the clauses that have been chosen, a firm can start out on a path and thus deprive itself of the possibility of modifying or adding certain clauses in order to make the license evolve.

The sources of this license dependence are factors which discourage the change in licenses. There are four of them.

The first is of a legal nature, in the sense that the same clauses of a license can prohibit registering the product under any other license without the agreement of all the joint authors, who are sometimes very numerous: this is the case of the GPL which stipulates that any GPL product or any product derived from a GPL product can only be registered under another GPL license.

The second source relates to the popularity of the product and its possible adoption as a standard: since a license can end up encouraging the adoption of a product, its transformation into a stricter version is likely, on the contrary, to lead users to turn to another product.

The third relates to institutions and lobbies, such as associations for the promotion of free software, such as the Free Software Foundation. These tend to encourage firms which edit standard products to register them under free licenses: any return towards a hybrid type of license would spark lively protests on their part.

Finally, the fourth relates to communication: changing licenses implies making significant communication efforts which are aimed not only at explaining the differences with the old version to the users, but also reassuring them about the consequences of such a change. This communication effort is all the more essential since the users are unreceptive and suspicious with regard to the motivations of the editing firm and they may even fear, for example, that it is reducing the free character of a license.

It appears, however, that the nuisances resulting from this path dependence are felt less strongly if the firm wishes to encourage the evolution of its license to a freer version. The opposite is true, however, if it tries to modify the license in the opposite direction (i.e. to limit the freedoms granted until then). Thus, in the case of Troll Tech, which chose to encourage the move from its hybrid license to a free version and then, finally, towards a GPL version, the path dependence effect was barely perceptible. On the other hand, one can imagine the case of a company having chosen a similar license for one of its products, the GPL or the LGPL, which are pure free licenses: the path dependence would have very harmful effects if it decided to return to a hybrid license, insofar as the 4 sources of path dependence referred to above would make it very difficult and dangerous to change the license.

8.4 Conclusion.

By basing ourselves on concrete cases, we have tried to show throughout this article that, for software editors, licenses are tools to conclude a predetermined strategy defined according to their position on the market. As such, the license seems to be a set of clauses, each of which serves as a precise strategic component. Take, for example, patrimonial valorization, the control of the competing firms, the development of co-operations which could lead to savings in R&D and standardization. This having been said, it is then possible to decipher each software license by updating the strategic orientations of the editor in accordance with the selected clauses. Going still further, it would be tempting to develop a methodology based on these observations which would make it possible to automatically associate each component of the firm's strategy to a specific clause of the license. Nevertheless, it would be a little hasty and perhaps even ineffective and dangerous over the long run. Indeed, if it is true that certain licenses are preferred according to the strategy adopted by the editor, one should not lose sight of the fact that strategies and licenses are opposed to one another on a precise point: their ability to evolve. If the license is characterized by its definite and static nature, this is not the case for the strategy to be adopted: it is very often led to evolve and change over the course of time according to the success of the product, the structure of the market or the competition. One thus understands the risks involved in adapting a static license to strategy that is constantly in flux: the consequence is a certain irreversibility and the negation of strategic opportunities. Consequently, for an editor, the choice of the license should no longer be limited to translating his/her current strategy into individual clauses, but rather request much more forward thinking and subtle work based on the anticipation of complex factors which take into account the entire life cycle of the product.

Part III

Studies of the juridical context of software economy.

the use of free software licenses.

YVES ROUGY,
Alcôve, Yves@rougy.net.

9.1 Introduction.

Free licenses are dominated by the GPL license which is the most widely used of all licenses at the present time. However, despite this supremacy, it should not be forgotten that other licenses also exist. Historically, the GPL license boom was a consequence of the democratization of the Linux operating system. Yet, older types of licenses, such as the BSD license or its by-products, are still common.

When a company develops a free software program, it is not unusual for a license to arise that is directly linked to the project or even to the company itself. Furthermore, these licenses often include the name of the company along with their own name.

Broadly speaking, it is almost always possible to associate a specific license to a specific project: Perl with the Artistic license, or BSD with the Unix system (Berkley University). This also shows that licenses do not always have to be part of a company.

Even though it is easy and fast to access these licenses through the Internet, it is particularly difficult to find an older version of a specific license or its historical background. Internet is so ephemeral that, for example, one page containing the evolution of a given license can disappear in a flash.

As a consequence, one can have access to older versions only if the author has put them on the website. Thus, the versions of the document are lost if the author has not filed them. Some information we would have liked to present in this paper is missing because we were not able to retrace the history of the licenses, such as the dates when they were created!

9.2 The License List.

The licenses presented here were found thanks to search engines, in particular the Google directory (http://directory.google.com).

Moreover, the information we found on the OSI site (http://OpenSource.org) was extremely helpful. As far as statistics relating to projects are concerned (thematic, distribution), we have only analyzed the most representative licenses. These statistics were based on the data found on the (http://SourceForge.net site. This site makes it possible for a programmer to host/place his project by specifying, among other things, the license which he/she has used. The large number of projects hosted (around 45,000), offers a sufficiently representative sample to study the licenses that have been used. Moreover, SourceForge divides its projects under different themes. For the sake of clarity, we have put all SourceForge themes together under more general themes. Over 35,000 projects come under this new classification. Although all the licenses are not listed on the SourceForge site, one can still find those that are the most representative.

9.3 Geographical Sources.

The geographical origin of the licenses was obtained by seeking the residence of the license creator. Usually, the information can be found on the same Website as that of the license.

From a statistical standpoint, 44 licenses were studied, [1]. The difference between this number and that of the total number of licenses stems from the fact that it is difficult to find the real origin of the licenses. Very often, the text and, eventually, the version of the license are the only elements which are available. Sometimes, it is information linked to the project which has led to the creation of the license that allows for the identification of the editing team which has worked on the license.

[1] the number of geographical sources is superior to 44 because several licenses result from international collaborations

The vast majority of licenses come from the United States: over 75 %. The rest come from Northern Europe (nearly 30 %). It is logical to notice that the most industrialized countries are those which produce the highest number of free licenses. Out of the 44 licenses we are dealing with, only one is the result of a collaboration between American, French and Japanese teams (the W3C license).

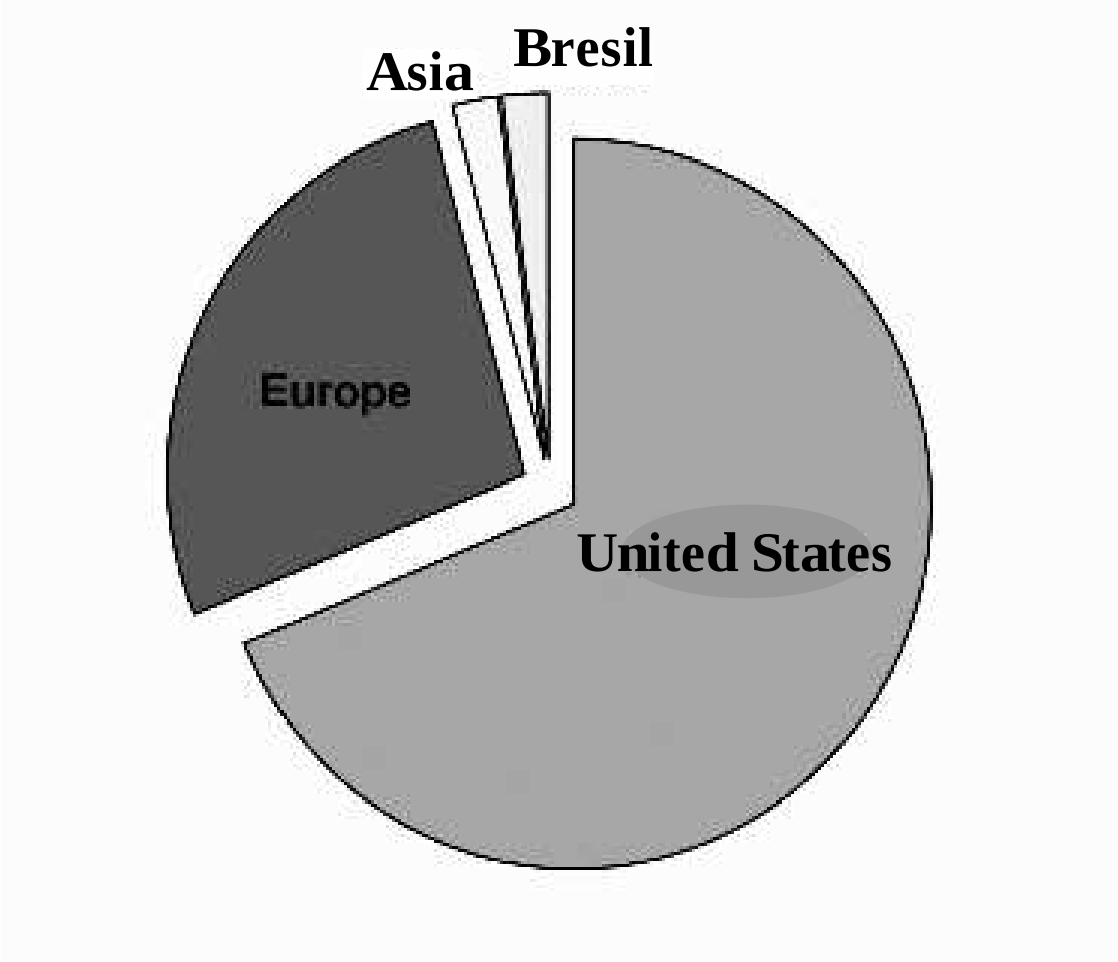


Figure 9.1 — Geographical distribution of licenses.

9.4 Free License Authors.

We have tried to classify the authors of the licenses we are studying into categories which we define as follows:

- Enterprise this is the case for licenses that are created by companies for a specific project or as part of an opening within the company's policy. Sometimes, creating a license is part of a given company's foundation.
- University: for universities and, in general, their research laboratories. They usually work in a very active way on innovative projects. As a consequence, quite a few licenses were created to diffuse such projects.
- Association: In this category one finds private individuals, foundations or associations that create free licenses such as the FSF, LaTeX, or company associations.

Sometimes, universities and companies work together as partners. So do companies and associations. This often happens within the framework of specific projects which contain an original license.

Once again, 44 licenses were studied to obtain these figures. Occasionally, there is not enough information to determine the origin of the license authors. Consequently, certain licenses are not included in the figures.

The majority of free licenses come from companies (22 licenses, 50 %). Universities come next with 15 licenses (34 %), finally, the associations with 10 licenses (22 %). Once again, the total figure is superior to 44, or 100 %, because the licenses that are used in partnerships are counted twice.

It is rather simple to understand such a distribution. Indeed, associations have a tendency to create widely used and generic licenses (like the GPL license, for example), whereas companies tend to create a license for each project. It is quite common to find one or more licenses linked to the same company. On the other hand, each university creates its own license and all university projects use the same license.

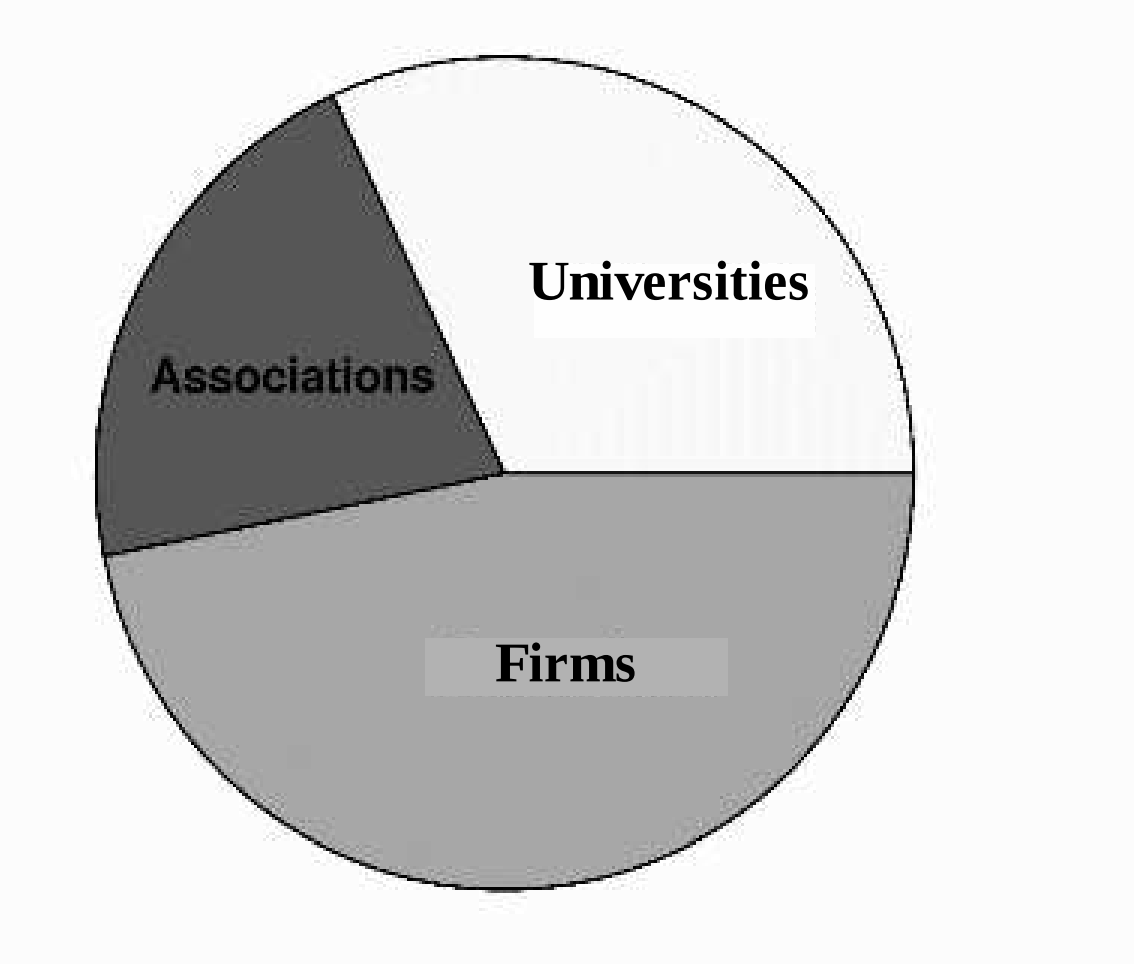


Figure 9.2 — Distribution of license origins.

9.5 License Distribution .

The GPL license is the most widely of all those represented since, out of the 45,000 projects that have been studied, almost 35,000 projects use it, that is to say, over 70 %. The LGPL license is next with more than 10 % of the projects —almost 5000 projects. The following license is the BSD type of license which represents

around 10 % of the projects. It has to be pointed out that this license is underestimated because many projects coming under a BSD type of license give it another name Then comes the Artistic license which is the Perl program license. Perl comes under both Artistic/GPL licenses. It amounts to less than 3 %. Finally comes the MIT license with a little more than 1.5 %.

As one can see, we are dealing with a rather unequal distribution. The FSF licenses (GPL and LGPL) represent more than 80 % of the projects that have been studied.

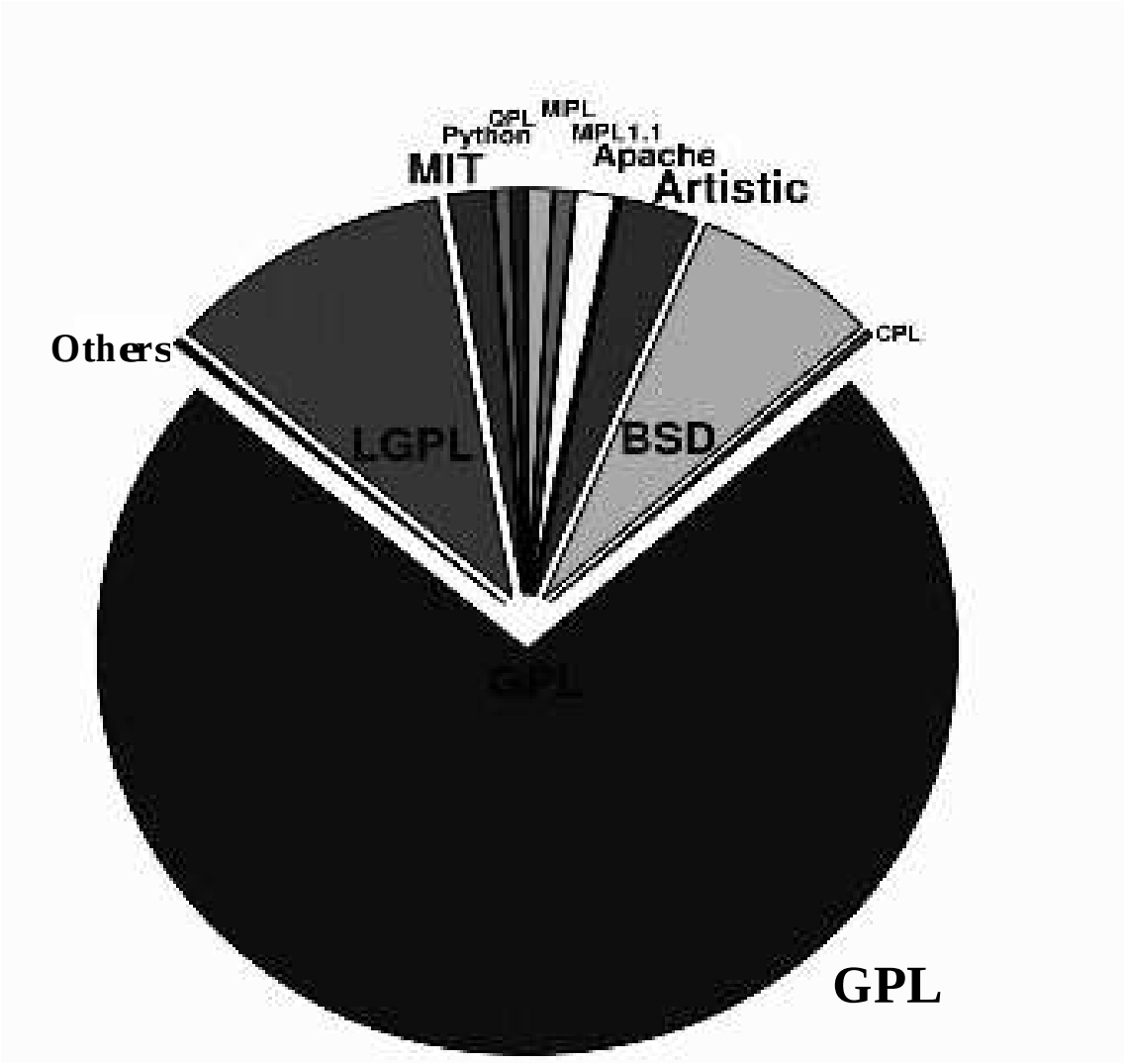


Figure 9.3 — Distribution of the projects according to the licenses.

9.6 Thematic Distribution of the licenses.

9.6.1 Introduction.

In determining these statistics, we wanted to see in which field(s) free licenses were mostly used. We also wanted to see which were the most frequent license themes.

This information cannot be found on sites which give the text of the license. It is thus necessary to count

the number of projects that come under a given license and to note the theme of each project. The source that allowed us to obtain this information was the community website SourceForge (http://sf.net).

9.6.2 General distribution.

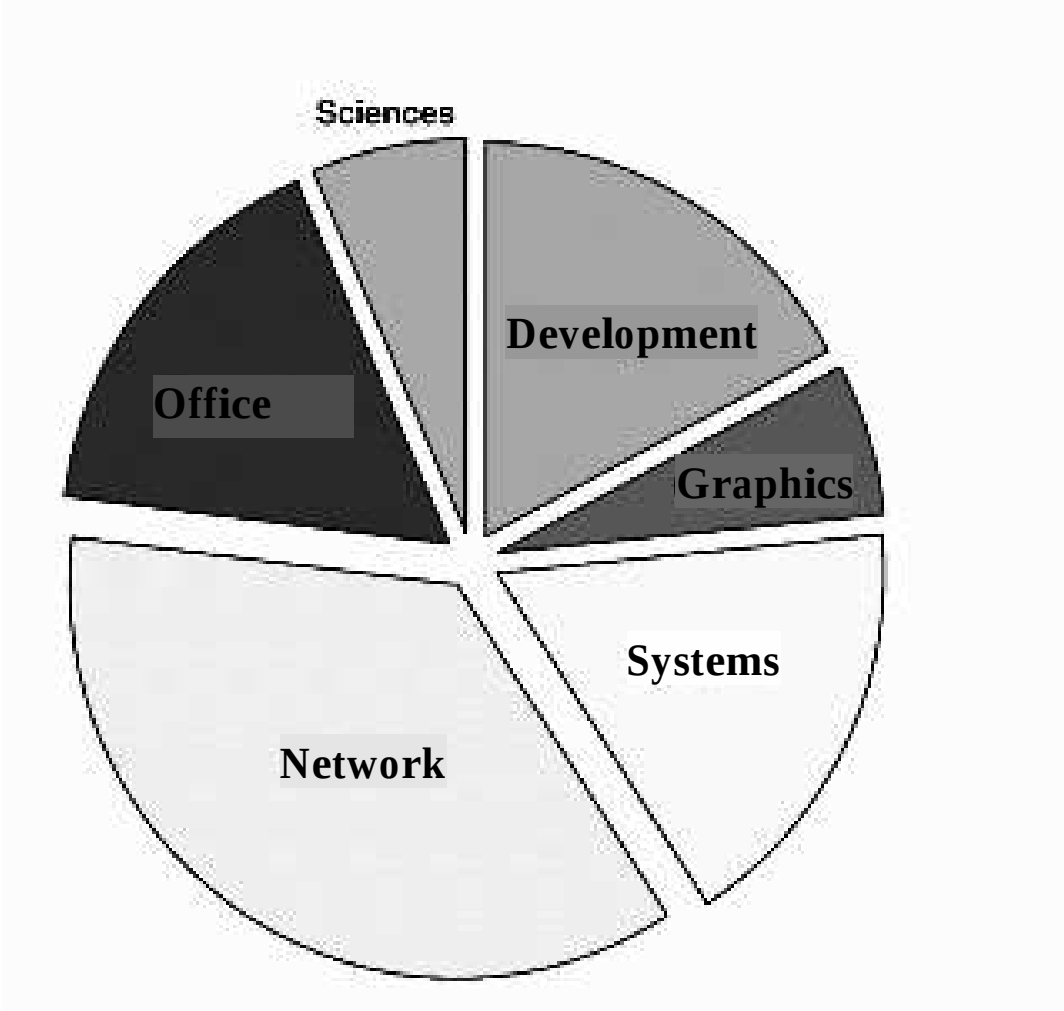


Figure 9.4 — General distribution according to themes.

Development	17,2 %
Graphics	6,2 %
System	17,4 %
Network	36,0 %
Office automation	16,6 %
Sciences	6,5 %

It is interesting to note that over a third of the free projects that have been studied deal with network applications. The popularity of free licenses has increased thanks to the Internet and also to Internet and network applications. The high number of projects linked to development or to the system shows how common it is to use free systems such as GNU/Linux.

On the other hand, the high number of projects which are linked to office automation is rather recent. Indeed, for the longest time, the absence of such free office automation projects was criticized. The strong presence of projects linked to the network and to the system also demonstrates the solid implantation of free projects in infrastructure applications.

9.6.3 The GPL licence.

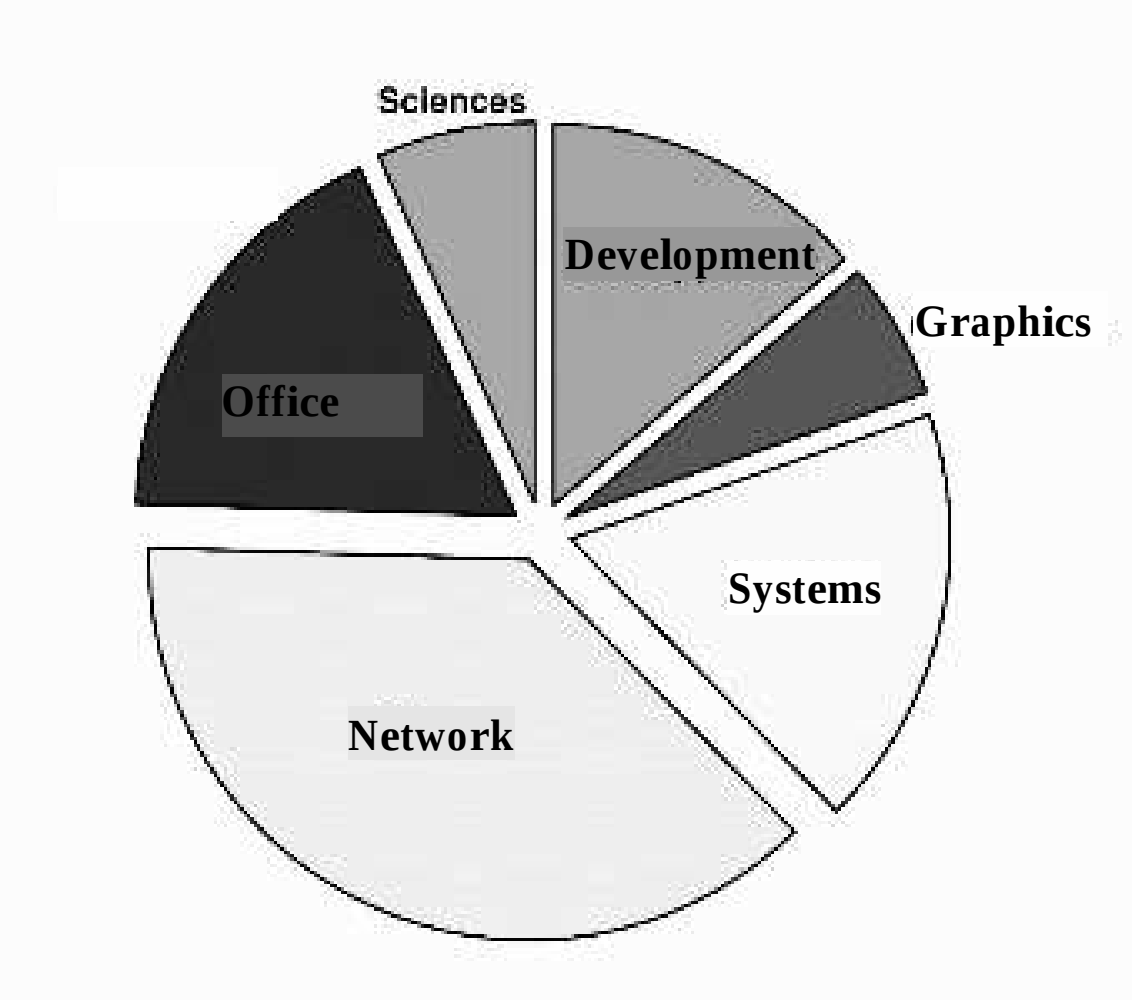


Figure 9.5 — Distribution of the GPL license according to themes.

Development	13,8 %
Graphics	5,8 %
System	18,0 %
Network	37,82 %
Office automation	17,8 %
Sciences	6,7 %

This distribution remains quite similar to the general distribution: the GPL license represents 75 % of the projects that were studied.

9.6.4 The LGPL license.

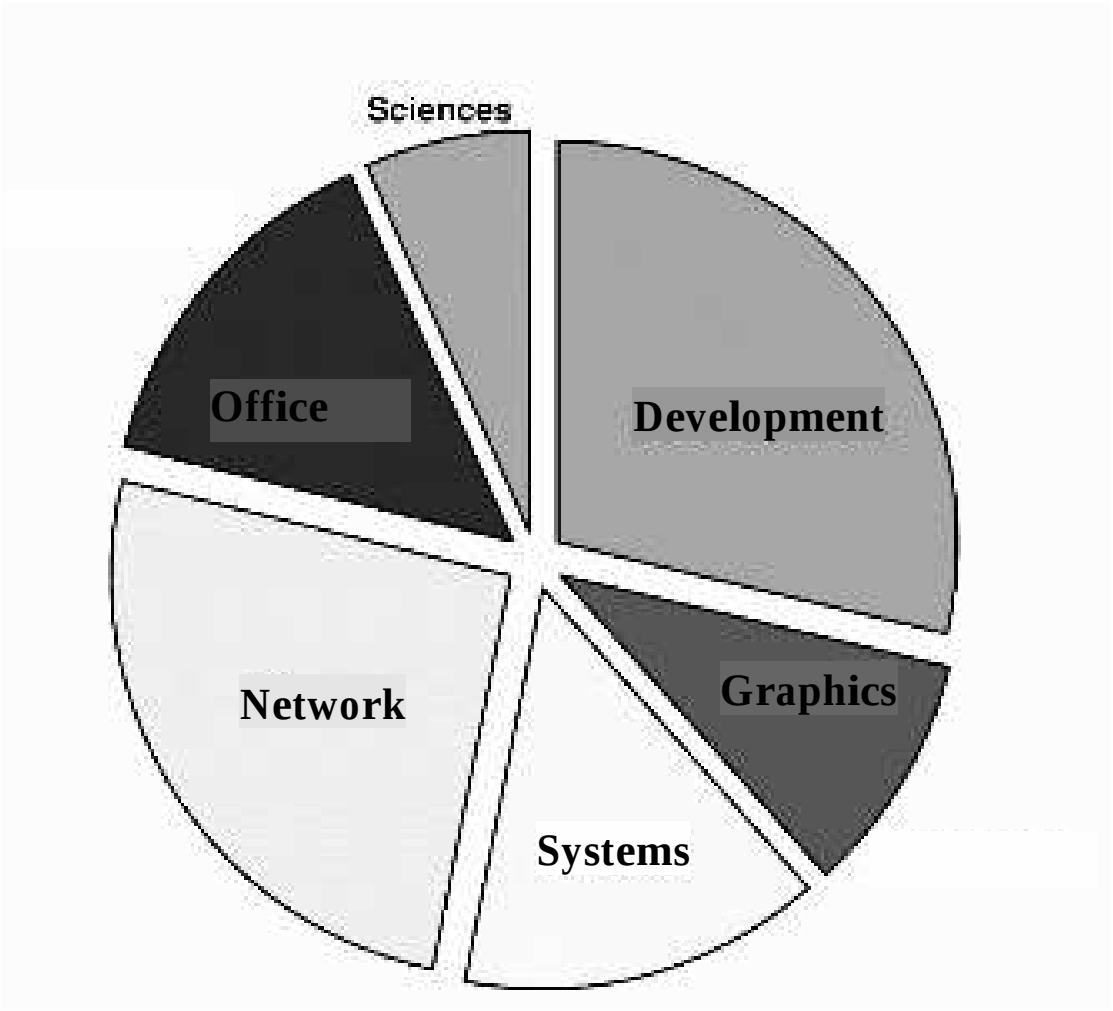


Figure 9.6 — Distribution of the LGPL license according to themes.

Development	28,7 %
Graphics	9,8 %
System	14,7 %
Network	25,7 %
Office automation	14,5 %
Sciences	6,7 %

Here, we observe strong involvement in development activities. Indeed, this license was written for programming libraries so that it could be used with software programs that were not compatible with the GLP license. This also seems to be the case for network projects.

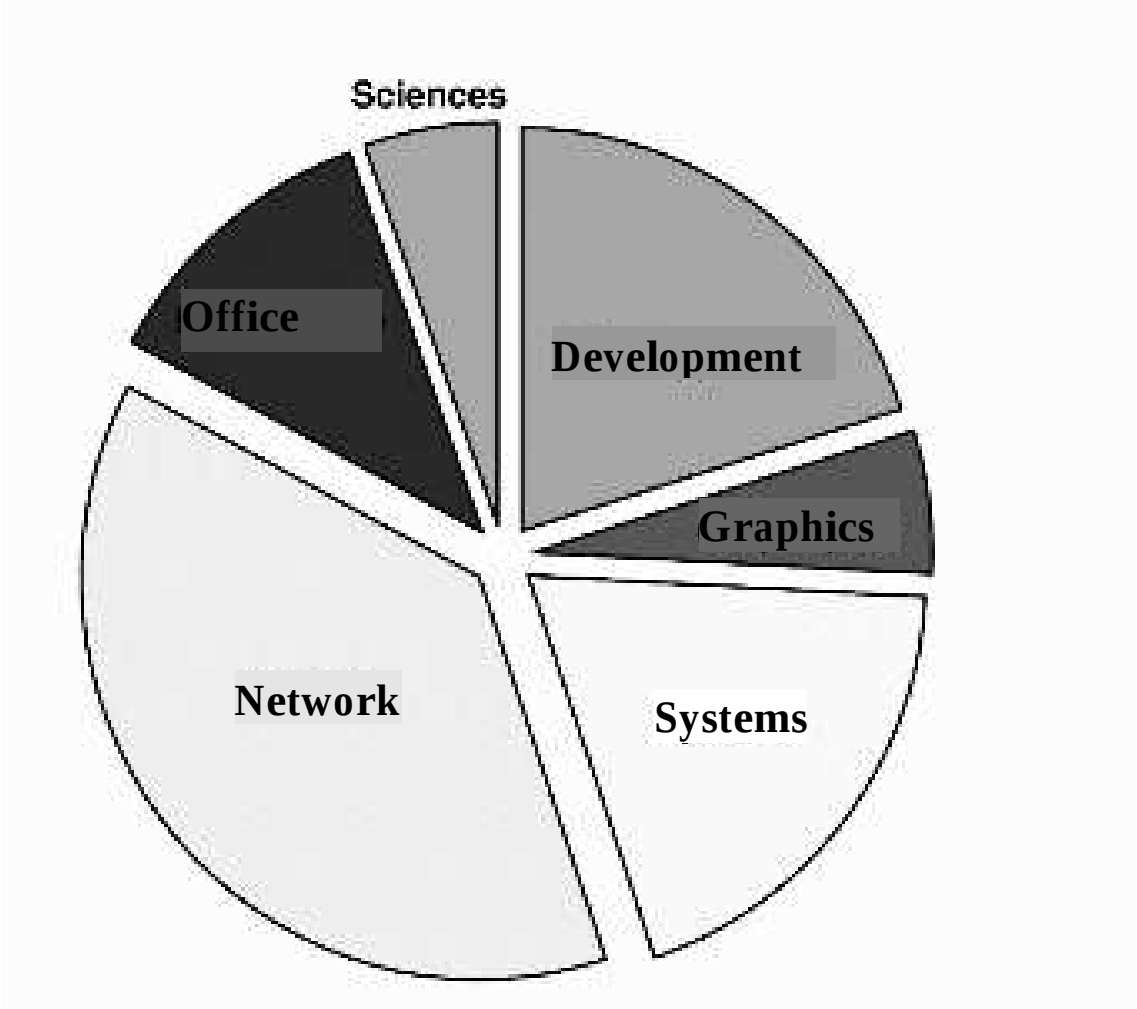


Figure 9.7 — Distribution of the BSD license according to themes.

9.6.5 The BSD license.

Development	20,1 %
Graphics	5,7 %
System	19,0 %
Network	38,0 %
Office automation	11,7 %
Sciences	5,4 %

The BSD license is strongly represented in network projects. The fact that the IP of the BSD systems was often re-used (this was permitted by the license itself), could have been a contributing factor.

Development and systems themes come next. The implication of these projects in the infrastructures is especially visible.

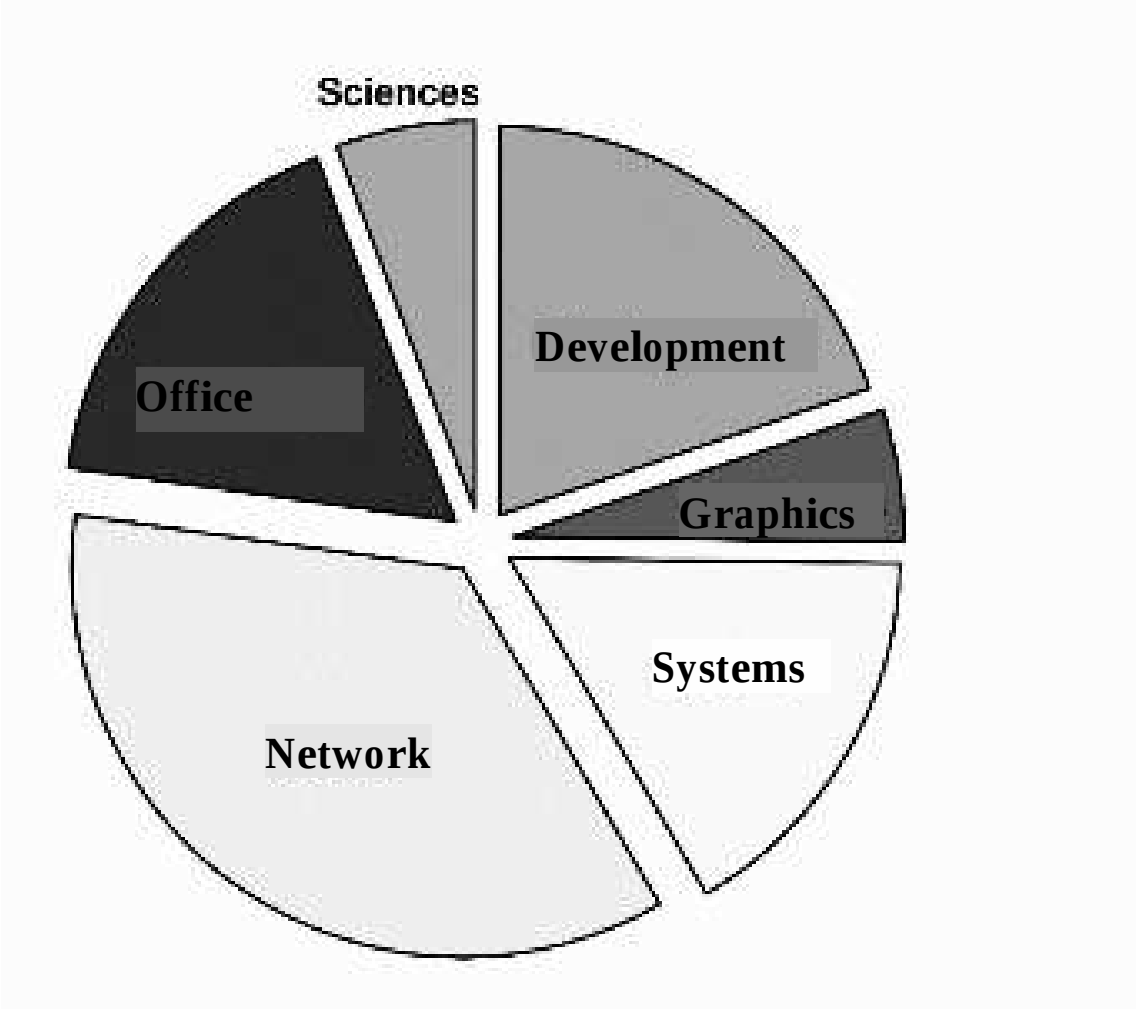


Figure 9.8 — Distribution of the Artistic license according to themes.

9.6.6 The Artistic license.

Development	19,7 %
Graphics	5,5 %
Systems	16,4 %
Network	35,6 %
Office automation	17,0 %
Sciences	5,7 %

The Artistic license is part of the Perl software program which is used in an extensive way in the Web and system applications. Moreover, Perl's modularity has allowed for the creation of a high number of projects which expand such capacities. The fact that the same license is used all the time may explain this distribution. Therefore, this license is extremely present in network and systems applications.

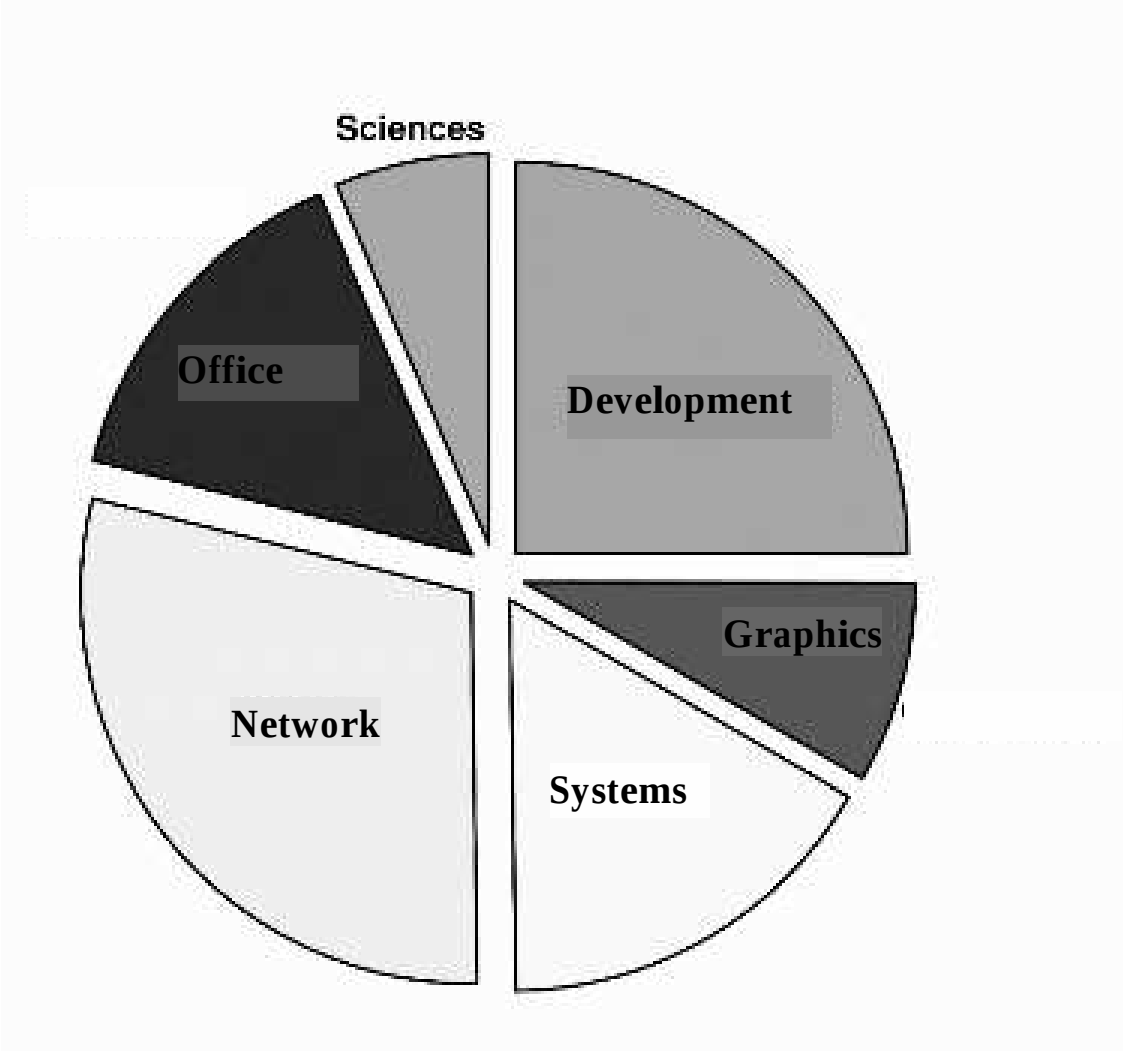


Figure 9.9 — Distribution of the MIT license according to themes.

9.6.7 The MIT license.

Development	25,0 %
Graphics	8,3 %
System	16,4 %
Network	29,1 %
Office automation	14,7 %
Sciences	6,4 %

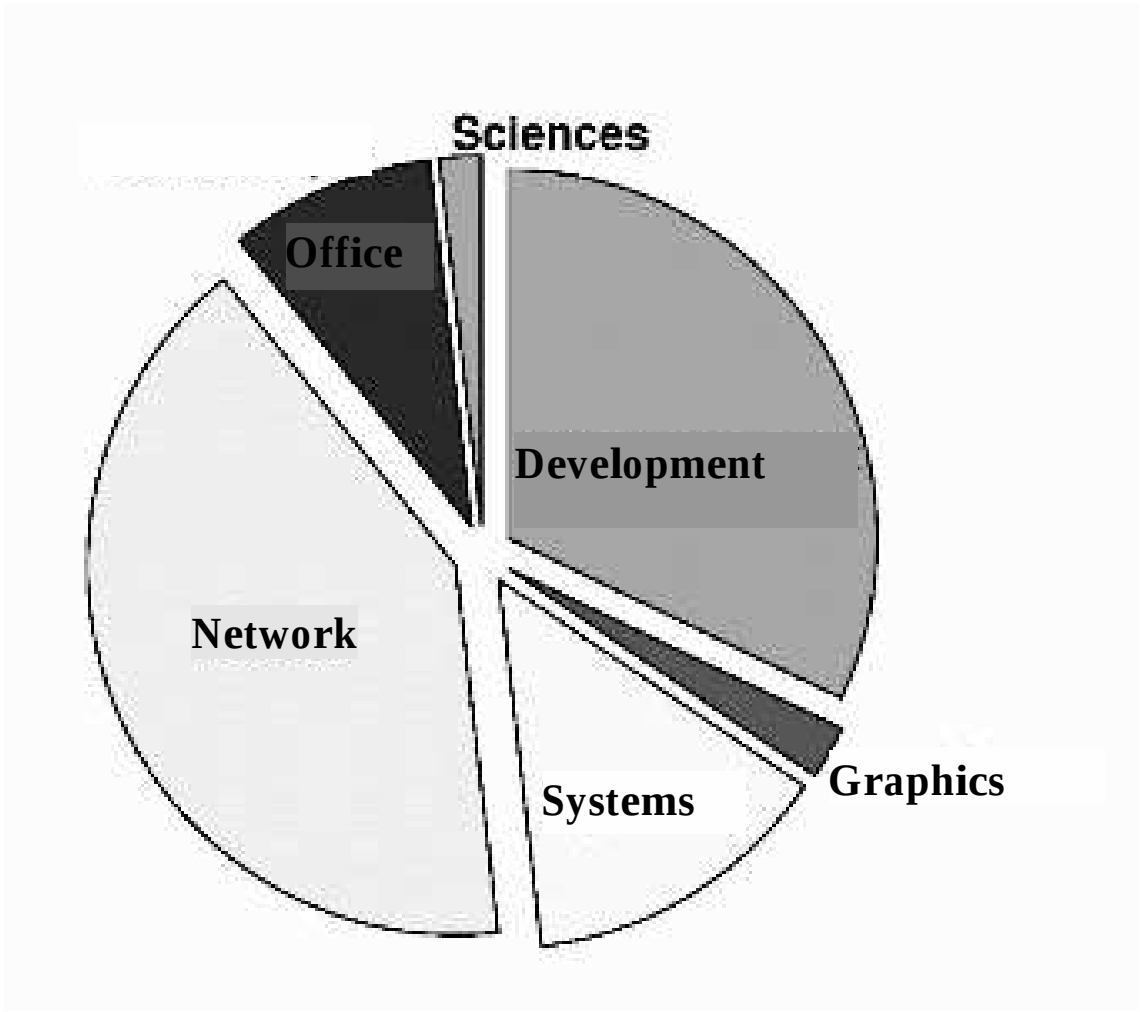


Figure 9.10 — Distribution of the Apache license according to themes.

9.6.8 The Apache license.

Development	32,1 %
Graphics	2,3 %
System	13,7 %
Network	41,0 %
Office automation	9,0 %
Sciences	1,8 %

This license concerns Apache software which is the most widely used Website server in the world. As a consequence, it is thus logical to see that over 40 % of the projects which use it are tied to network activities.

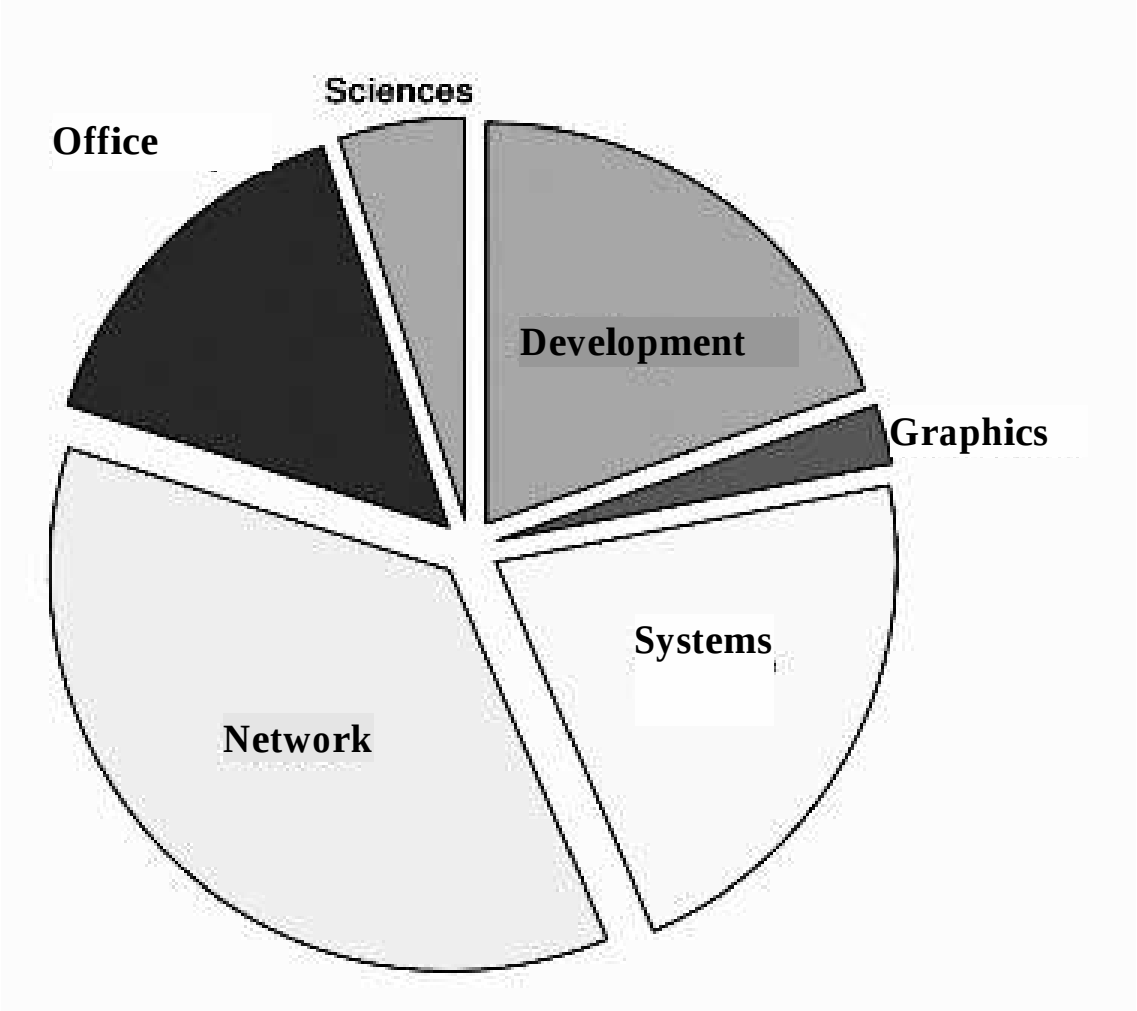


Figure 9.11 — Distribution of the MPL license according to themes.

9.6.9 The MPL license.

Development	19,5 %
Graphics	2,4 %
System	21,7 %
Network	36,4 %
Office automation	15,0 %
  Sciences	5,1 %

This license is the first license under which the browser Mozilla was published. The weaknesses of this license appeared quickly enough and a 1.1 version was created (cf. infra.). However, many projects are still published under this license, not under the 1.1. version.

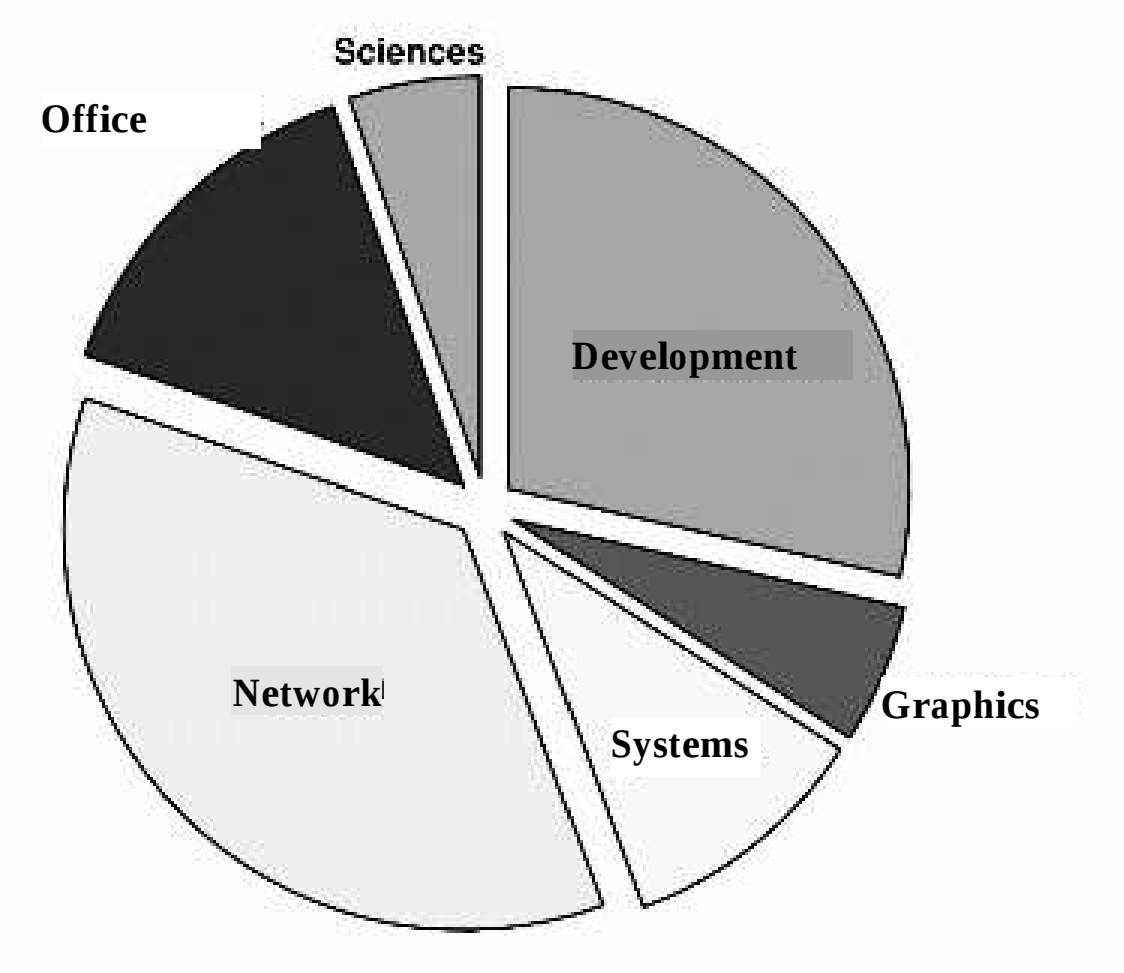


Figure 9.12 — Distribution of the MPL 1.1 license according to themes.

9.6.10 The MPL 1.1 license.

Development	28,5 %
Graphics	5,6 %
System	10,2 %
Network	35,9 %
Office automation	14,4 %
Sciences	5,3 %

9.6.11 Other licenses.

Each one of the other licenses that have been studied represents less than 1 % of the total number of licenses. That is why we have chosen to bring them together in one study.

- APSL (Apple): The main themes encountered concern the network, development and systems activities.
- CPL: This license, which is widely used by IBM, mostly concerns development activities.

- IPL: This license has allowed the creation of CPL. It mainly concerns network, development and system activities.
- Eiffel: This license which is associated with the eponymous language, mainly concerns development activities.
- Intel: Mainly concerns network activities.
- Jabber: This license was introduced through a project tightly linked to network activities. It is thus logical that such activities are predominant under this license.
- Python: The license of the eponymous language* is equally divided between development activities and network activities (50 %).
- QPL: This license, which is linked to “Qt”; display libraries, is split between development activities, but also network activities. Because the number of Unix type machines for office automation has risen and because the KDE environment has matured, we can find a rather high number of projects which concern office automation.

Free licenses. Legal Analysis.

MÉLANIE CLÉMENT-FONTAINE,
Alcôve, University of Montpellier I/ERCIM, melanie@amberlab.net.

10.1 General Definitions.

10.1.1 License.

The license is a convention whereby a person (called the license donor) authorizes another person (called the licensee) to use his/her software without transferring the rights which are attached to it.

10.1.2 Software.

Both the 1991 Community directive[1] and the WIPO (World intellectual Property Organization) clauses speak of a program which they define as "a set of instructions which, once they are transposed through a medium which is decipherable by a machine, can indicate, realize or obtain a function, a task or a specific result thanks to a machine capable of processing information" (WIPO standard clauses relating to the protection of software). However, "logiciel" (i.e. software) is the term that has been retained in France. Moreover, it covers the "description of the program" and its "auxiliary documentation". In addition, both the directive clauses and what is known as intellectual property under French law (article L. 112-2 13 of the CPI) include "preparatory design hardware"[2]. The software is described as intellectual property and, as such, is protected by copyright (see supra: Preliminary chapter).

Distinction between specialized software and software packages:

- *specialized software*: the client makes an order from the service provider so that the latter can design and realize a program that is adapted to his/her needs.
- *software package*: "a complete and documented set of programs conceived on behalf of several users for the same application or functions." (Ministerial order on the improvement of computer terminology, Dec. 22, 1981, JO[3] Jan. 17 1982, p. 624).

10.1.3 Open Source.

The Open Source Initiative offers eight criteria to identify the free licenses which are described in the Open Source Definition documenthttp://www.gnu.org/philosophy/free-sw.html:

1. the license must not make obtaining royalties a precondition in exchange for access to the software. However, this does not necessarily mean that no royalties can be requested either;
2. the license must allow users to have access to the source code;
3. the license must allow the user the right to modify and distribute derivative works in accordance with the conditions set forth in the initial software license;
4. the license, while encouraging modifications and thus the evolution of the programs, must guarantee the paternity of the author. For that reason, it can happen that the modifications made to the software can only be distributed in the form of "patch files" (corrective files). In addition, the license must expressly allow the distribution of the combined software deriving from the modified source code. Finally, it might be obligatory for derivative works to have names or numbers which differ from those of the original software version;

[1] Cons. Dir. EC n°91/250, may 14, 1991, JOCE may 17, n°L122, p. 42.

[2] Vivant [1997], p. 51.

[3] The JO, “Journal Offi ciel”, is the French government’s offi cial bulletin in which new laws, civil service jobs, etc. are published daily.

5. the license must not allow discrimination against people, groups or contracts in certain spheres of activities (in particular commercial activities);

6. the distribution of the software should not be subjected to additional requirements such as a "non-revelation" agreement;

7. the authorization should not be limited to a particular use;

8. it is not necessary for the software programs that go with the one subjected to the license to be open source.

10.1.4 Copyleft.

The Free Software Foundation considers that a license is free insofar as it offers the user the following[4]:

- the freedom to execute the program for any type of use;
- the freedom to study how the program works and adapt it to one's personal needs. This of course, means that it is necessary to have access to the source code;
- the freedom to redistribute copies;
- the freedom to improve the program and to publish these improvements. Here again, it is necessary to have access to the source code.

10.1.5 "Free copy" licenses.

The software that comes under the term "free copy" are subjected to licenses which do not make it possible to modify them. This means that "free copy" licenses are not really free licenses.

License freeware.

It allows the general public to use the software programs for free.

License shareware.

It is a contract whereby the author of a software package authorizes its reproduction totally free of cost. This allows the users to test it before buying it. This is done over a predetermined period of time at the end of which the user can decide whether or not to keep the software. If this is the case, the user will pay royalties.

License crippleware.

It allows for the use of a software under a version which will only be functional during the trial period.

[4] http://www.gnu.org/philosophy/free-sw.html

10.1.6 Free licenses Certification.

In the case in point, the certification consists in determining whether a license is free or not.

The Free Software Foundation (FSF) and the Open Source Initiative (OSI) are two organizations which are recognized by the free software communities in order to validate free licenses. Each organization has defined certification criteria:

- for the Open Initiative Source (see supra 1.3),
- for the Free Foundation Software (see supra 1.4).

Open Source Initiative.

The OSI has set up an attestation procedure to determine whether or not a license can be officially declared to be an Open Source license[5].

To submit a license for validation, it is necessary to write to: license-approval@opensource.org.

Once the license has been approved, the software that comes under this license can be called an "OSI Certified Open Source Software."

Free Software Foundation.

The FSF system is a little less formal. It offers a list of licenses with a commentary about each one of them[6]. The FSF classifies the licenses according to the following key points:

- is the license a free software license or not?
- is the license a copyleft type of license or not?
- is the license compatible with the GNU GPL (i.e. can one combine a module under GNU GPL and another one under another type of license to obtain a larger module)?
- does the license pose specific, practical problems?

The FSF has a list that makes it possible to evaluate any type of license and to answer any type of question, particularly about the GNU GPL[7].

10.2 General Analysis.

Free software licenses have general characteristics which definitely raise questions from a legal point of view.

10.2.1 The language.

Presentation.

Until now, not one free software license has been written in French. They are all written in English. That is primarily due to their origin (see Yves Rougy's study on the use of free licenses). One might also think

[5] http://www.opensource.org/licenses/index.html
[6] http://www.gnu.org/licenses/license-list.html
[7] licensing@gnu.org

that the English language is better adapted for the distribution of software programs which reach a worldwide audience. If the context in which such programs are created and the practical contingencies they entail justify such a choice, is it valid under French law?

Legal analysis.

The main texts that relate to the obligatory use of the French language are as follows:

- August 4, 1994 n° 94-665 law, known as the "Toubon Law[8]", relating to the use of the French language (JO. April 5, 1994), which revokes and replaces the December 31, 1975 n° 75-1349 law (JO. January 4, 1976).
- Constitutional Council Decision n° 94-345 July 29, 1994 (JO. of August 2, 1994);
- March 3, 1995 decree n° 95240 concerning the application of law n°94-665 of August 4, 1994 relating to the use of the French language (JO. March 5, 1995);
- March 19, 1996 circular concerning the application of law n°94-665 of August 4, 1994 relating to the use of the French language (JO. March 20, 1996);
- March 6, 1997 circular relating to the use of the French language in the information and communication systems of State-run administrations and public institutions (JO. March 20, 1997).

The use of the French language is compulsory in:

- "the designation, the offer, the presentation, the instructions concerning the use of goods, products or services. This also applies to their warranty (how long does it last, what are the conditions, etc.) as well as invoices"[9];
- this obligation "does not apply to foreign products and specialities which are known internationally"[10];
- contracts "whatever their subject or form", "to which legal entities or private individuals carrying out missions recognized as being beneficial for the general public are parties"[11].

The law can be applied to both private and public individuals. However, some of these clauses are more constraining for individuals who carry out missions which have been recognized as being beneficial to the general public[12].

As far as contracts are concerned, the use of the French language seems compulsory only when a legal entity or a private individual carries out missions which have been recognized as being beneficial for the general public. Moreover, the clauses of article 2 of law n°94-665 of August 4, 1994 seem to apply only in the event of "the marketing of goods, products and services[13]." Finally, "the invoices and other documents exchanged between French or foreign professionals under private law, who are not the final consumers or users of goods, products or services[14]", do not necessarily have to be written in French.

[8] Jacques Toubon was the right wing Minister of Culture from 1993-1995 who wrote a controversial law making the use of French obligatory law in a number of specifi c contexts.

[9] Article 2 al. 1 of law n° 94-665 du of August 4, 1994.

[10] Article 2 al. 2 of law n° 94-665 du of August 4, 1994.

[11] Article 5 of law n° 94-665 du of August 4, 1994.

[12] Article 2 of the March 19, 1996 decree.

[13] Article 2.1 of the March 19, 1996 decree.

[14] Article 2.1,§2 of the March 19, 1996 decree.

Sanctions are listed in Decree n° 95240 of March 3, 1995. The non-observance of these clauses is punishable under the law. The fine is the same as those belonging to class 4 violations.

This shows that it is not compulsory to use the French language in all instances. Public individuals are those who are most concerned by this restriction.

10.2.2 Obligation of copyright registration.

Principal texts:

- Law n° 92-546 of June 20, 1992 relating to the registration of copyrights (JO. June 23, 1992);
- Decree n° 93-1429 of 31 December 1993 relating to the registration of copyrights (JO. January 1, 1994).

The registration of copyrights is required:

- for "software packages, data bases, expert systems and other artificial intelligence products that are subjected to legal obligations as soon as they are put at the disposal of the public through the diffusion of any hardware support, regardless of the nature of this support"[15]. The form of diffusion generally adopted for free software is on-line diffusion. Thus the requirement of copyright registration is not applicable in the majority of cases. However, it occasionally happens that free software is diffused in the form of a paying CD-Rom and, most notably, with a complete instruction guide and a maintenance contract[16]. In this case, people who edit, produce or import software packages are obligated to register the copyright[17]. It should be noted in a report that the Council of State has recommended amending the law in order to extend this obligation to cases where software packages are made available to the public through a numerical "open" network[18].
- for the software packages, data bases, expert systems and other products considered to be representative of artificial intelligence by decision of the Industrial Culture and Research Ministry, based on proposals by the Advisory Commission and upon consultation with the Scientific Council on the registration of copyrights. These decrees are published in the Journal Officiel. The copyright must be submitted within eight days following its publication. Absence of registration does not affect copyright protection, but is legally sanctioned with a fine[19]. In the case in point, this obligation is consequently limited to a small number of software programs.

10.2.3 The Version.

Presentation.

The most elaborate free licenses are the subject of a clause providing that new versions of the License could be published in the future (examples: Mozilla Public License version 1 and Netscape Public License, article 6; Public IBM License version 1.0, article 7 & 4; General Public License version 2, article 9; APPLE Public Source License version 1.2, article 7; Sun Community Source License, version 3, article V A). Only the organization that has proposed the license is entitled to create a new version which will have a distinct number.

[15] Article 1 of Law n° 92-546 of June 20, 1992.
[16] The companies RedHat, MandrakeSoft, Caldera or SuSE thus distribute the various versions of Linux.
[17] Article 4.3°. law of 1992.
[18] Les études du Conseil d'État [1998]
[19] Article 7, law 1992

Exemples:

APPLE public source license: 7. Versions of the License. "APPLE may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Once Original Codes has been published under a particular version of this License, you may continue to use it under the terms of that version. You may also choose to use such Original Code under the terms of any subsequent version of this License published by APPLE. No one other than APPLE has the right to modify the terms to Covered Codes created under this License."

Netscape Public license: 6. Versions of the License. "6.1. New Versions. Netscape Communications Corporation ("Netscape") may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number. 6.2. Effect of New Version. Once Covered Code has been published under a particular version of the License, you may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Netscape. No one other than Netscape has the right to modify the terms applicable to Covered Code created under this License."
The clause is identical in the Mozilla license.

Public IBM License: article 7.

"(...) IBM may publish new versions (including revisions) of this Agreement from time to time. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributors may elect to distribute the Program (including its Contributions) under the new version. No one other than IBM has the right to modify this Agreement. (...)"

Consequences of a new version. In general, the licenses specify that the user has the choice between the terms of either the new version or of the original version of the software.

The General Public License version 2, article 9 distinguishes according to the following assumptions: if the software indicates a version NUMBER as well as the declaration stating "any later version", then the user can apply the clauses mentioned in the indicated version or in any other more recent version published by the Free Software Foundation. If the software does not specify any version number, then the user can choose any of the versions published by the Free Software Foundation.

Legal analysis. A party cannot envisage modifying the terms of an agreement unilaterally even if the modifications are only minor ones and would not change the spirit of the license[20]. Indeed, the user has consented to respect the conditions foreseen by the license.

However, the modification of the license agreement is not unilateral, insofar as it is up to the user to accept the new version or, on the contrary, to refuse it and continue to respect the terms of the initial version. Thus, to be opposable to the licensee, the modification of the license must be the decision of all parties. Only under such conditions may these clauses be legally acceptable.

Lastly, it is perhaps worth noting that new versions are rather rare. The free software licenses are, globally speaking, stable.

10.2.4 Clauses relating to the warranty and liability.

All free software licenses contain an elusive clause of liability.

[20] The GPL, for example, specifies that the new versions will be similar to each other. In other words, from one version to the next, only some details differ in order to solve new problems.

Examples.

Mozilla Public License version 1 and Nescape Public License. “7. DISCLAIMER OF WARRANTY

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE GOLD NON-INFRINGING. THE ENTIRE RISK WITH REGARD TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE RESTS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.”

“9. LIMITATION OF LIABILITY

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES? BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, COMMERCIAL OR ANY AND ALL OTHER DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH A PARTY’S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THAT EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU."

IBM public License version 1.0. “5. NO WARRANTY EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damages to or loss of data, programs gold equipment, and unavailability or interruption of operations."

See also Public APPLE Source License version 1.2, article 8 AND 9; Sun Community Source L I cense, version 3, article V B and C, ...

Legal analysis.

Can one thus justifiably push aside any warranty and all liability?

This question is not a new one and predates the development of free software licenses. The Commission responsible for abusive clauses considered the problem sufficiently alarming to submit a recommendation on April 7, 1995 "relating to the contracts proposed by the editors or distributors of software or software packages

intended for use on personal computers.[21]"

On this occasion, the Commission identified eight contractual clauses relating to the warranty which it considered to be abusive from a legal standpoint. Indeed, it notes that "these contracts are worked out by professionals and the use of software generally requires that the consumers respect these clauses(...) These are thus contracts that are usually proposed by professionals to consumers". The rules concerning consumer law are thus applicable to them. The Commission recommends, in particular, "the elimination of the clauses from the contracts under consideration whose goal or consequence is: (...)

1° to exclude any professional warranty relating to the software, its support and to exonerate it from all the consequences of flaws in the documentation provided at the time of the provision of the software;

2° to lead the consumer astray by combining provisions which exclude all warranty with restrictive warranty clauses;

3° to exonerate the professional of any liability due to detrimental consequences stemming from the use of software programs which he/she markets"(...)

Since it is a mere recommendation, this text has a limited scope. However, it has an instigating influence and can be used as reference by a judge. One can wonder whether all the free software licenses are affected by this text insofar as the Commission focuses only on those contracts drawn up by professionals and that free software is not always issued by professionals. Most especially, there is not any doubt that the contracts concerned here are those used in the marketing of software. In 1995, new economic models, such as those which are the subject of this RNTL (National Network for Research in Software Technology) report, had not yet been published and free licenses were known only to specialists. However, this new type of license does not represent the same power struggle between the distributor and the user. Indeed, one might wonder whether the nature of free licenses, and especially the extent of the freedoms offered to the users, should not lead to other conclusions than those applied to traditional licenses. Legislation concerning the protection of the consumer tends to maintain a balance between the parties by coming to assistance of the consumers who are considered to be in a position of weakness. Yet, if the Commission could consider that those ambiguous liability clauses in the traditional licenses create a significant imbalance, balance, on the other hand, comes from quite a different source, namely, free licenses. Thus, the absence of a warranty could be justified in the case of free licenses, in our opinion, on account of the freedoms granted to the user, in particular that of modifying the software without paying a single red cent.

Nevertheless and notwithstanding what has just been stated, the May 19, 1998 law[22], transposing the directive on faulty products[23], the obligation to make reparations for damages done to persons or to goods, other than reparations to the faulty product (article 1386-2 C. Civ.) rests on the manufacturer (article 1386-6 C Civ). It follows from these provisions that there exists a general principal by which the manufacturer is held responsible for any defects in the products he/she puts on the market. Furthermore, the right to reparation concerns not only the purchaser of such products but also third parties. These solutions apply to the whole of the European Union. The definition of the product aimed at by the law is broad (article 1386-3 C Civ.). Due to this fact, the concept seems to include software as soon as the latter appears to be defective (article 1386-4 C Civ.). In order to obtain reparation, one only need prove "the damage, the defect and the bond of causality between the defect and the damage" (article 1386-9 C Civ.). A "product is considered to be defective (...) when it does not offer the safety which one can legitimately expect" (article 1386-4). This liability only ceases "ten years after the product that has caused the damage has been put on the market" (article 1386-

[21] Recommendation n° 9502, BOCCRF August 25, 1995.

[22] Law n° 98-389, May 19, 1998, JO May 21, 1998.

[23] Directive n° 85/374/EC, relating to the bringing together of the legislative, regimental and administrative clauses of the Member States regarding liability whenever a product is defective/faulty, [European Communities JO, n° L210, August 7, 1985.]

16). The law extends the liability of the manufacturer to those who rent the product as well as all other professional suppliers." (article 1386-7 C. Civ.). These provisions could apply to the license donor as soon as it concerns a professional supplier. Thus, the independent developer who is not a software provider, should not be worried by this heavy responsibility. On the other hand, the professional supplier cannot be exonerated from his responsibilities. Any clauses stating the contrary are considered null and void * except if the clause is stipulated between professionals or if the damage has been caused to goods whose use or consumption is not primarily private (1386-15 C. Civ.). In such a case, he/she can only meet the criteria for cases of exemption under exceptional circumstances outlined in article 1386-11 of the Civil Code.

10.3 Attempt at classifying free software licenses.

It has become commonplace today to differentiate between three types of free software licenses: those known as public domain licenses, commercial company licenses and copyleft licenses. This classification rests primarily on the effects of the licenses and secondarily on their origins.

10.3.1 Public domain licenses.

The best known are Berkeley's BSD license, the Apache license (Apache Sofware License), the Mozart license (License Agreement for Mozart), the Cryptix license (Cryptix General License), and the MIT license. These licenses have two things in common: they are very short and have a declaratory tone. Three items are discussed by these licenses: authorized uses, conditions of use and warranties.

Authorized uses.

The authorized uses are those that are typical of free licenses:

- authorization to use the software;
- authorization to modify the software;
- authorization to distribute the modified software or not.

Conditions of use.

The conditions of use of these free licenses are not very constraining.

Mention of the copyright
The example of Apache Sofware License:

- the distribution of the source code must preserve the mention of the copyright, the list of the conditions of each authorization and the discharge of liability;
- the distribution of the binary form* of the software must reproduce these same indications in the document and/or the other materials provided when it is distributed;
- If there is documentation attached to the distribution, it is advisable to indicate the following formula: "This product includes software developed by [name of the organization] (http://www.apache.org /)";
- this recognition can appear in the software itself, and everywhere where such acknowledgements by third parties normally appear.

The protection of the name of the organization at the origin of the license.

These licenses are written by organizations, like the University of Berkeley or MIT, to accompany the software which they develop in-house. They are not intended to be used for the needs of third parties who want to diffuse their own software. However, the developers can inspire themselves from them in order to write their licenses. One can note that the practice of copying existing licenses almost word for word while simply changing the name is very widespread. Nevertheless, this is not any less reprehensible with regard to copyright violation. Indeed, because a license is original, the author is protected from copyright infringement under the law.

The fact that these licenses are written for the specific needs of the beneficiaries of these rights (who usually are legal entities) means that their name is present in the title of the license and that their copyright is indicated: for example, "Copyright (c) 2000 The Apache Software Foundation. All rights reserved".)

If the use of the software subjected to such a license is free, the name of the organization should not be used for promotional purposes. In this manner, the organizations can protect their name and their reputation.

The non-warranty clause.

This subject has already been discussed (see supra 2.4).

10.3.2 Commercial firm Licenses.

Certain distributive software firms have decided to simultaneously develop free (more or less) software. These firms have written their own licenses. But, this phenomenon remains rather rare.

The choice of the companies to use their own licenses is not explained solely by their concern for recuperating their own license clauses but, also, by the desire to avoid harming their commercial software distribution activities which they have not abandoned.

Traditional editing.

Commercial firm software licenses use licenses which are edited in a traditional style insofar as one encounters the standard clauses one finds in all contracts:

- duration of the contract,
- clause indicating the law applicable to the contract,
- clause indicating the court of jurisdiction in the event of litigation...

The presence of these clauses not at all unusual because they are so standard. Yet, when they show up in free licenses, their presence can be surprising because, in this framework, they are so rare. They only appear in commercial firm licenses.

One can imagine several reasons for this phenomenon: the writers of commercial firm free licenses are the same as those who write other proprietary licenses. They thus reproduce what are accustomed to writing by applying models they are familiar with. In addition, even if these clauses might be highly useful when one knows the identity of the license donor, as is the case for commercial firm licenses, they are very often useless in the other contexts. Thus, since a license like the GNU GPL license can be used by any software author regardless of his nationality or his place of residence, the fact of indicating the applicable law or the competent jurisdiction will very often be a waste of time.

The Conciliation of Free Distribution and Proprietary Distribution.

In order to reconcile the two modes of software distribution, one with the conditions of free licenses and the other with the conditions of proprietary software, commercial firms frequently reserve certain prerogatives vis-à-vis the users. Moreover, they confine the freedoms to the rights attached to software to the exclusion of other intellectual rights relating in particular to a trademark or a logo. Such protectionist measures are sometimes perceived to be poles apart from the free software philosophy. In order to confirm the free software image of certain projects, commercial firms sometimes adopt a distinct license which does not contain such restrictions.

The Reservation of Prerogatives.

Netscape Public License version 1.0 presents a very good example regarding the reservation of certain prerogatives in order to reconcile free software distribution with proprietary software distribution. The license organizes a system which is specific to Netscape. Indeed, if the modifications made to the software are, in theory, governed by the terms of the license (article 3.1), the Netscape company can, if it so desires, break away from these obligations: it reserves the right to include the software in one of its non-free products during the two years following the date of the initial version of the code with an additional proviso that it not be subjected to the conditions stipulated in the license.

For its part, the APPLE company reserves the freedom to choose whether the modifications it makes or which are made on its behalf will be subjected or not to the conditions stipulated in the APPLE public license or to other conditions. One may wonder if contributors who have made modifications to the initial software can also benefit from such options. The clause is not very clear on the matter: initially, it is specified that, both the contributors and APPLE maintain their rights with regard their contributions and, subsequently, it specifies that APPLE is free to choose how to distribute the modifications that have been made without mentioning the other contributors. It seems as if APPLE does not wish to extend such freedom of action but yet, does not clearly state it either.

“11. Ownership. Subject to the licenses granted under this License, each Contributor retains all rights, title and interest in and to any Modifications made by such Contributor. APPLE retains all rights, title and interest in and to the Original Code and any Modifications made by or on behalf of APPLE ("APPLE Modifications"), and such APPLE Modifications will not be automatically subject to this License. APPLE may, at its sole discretion, choose to license such APPLE Modifications under this License, or on different terms from those contained in this License or may choose not to license them at all."

Excluding the other rights of intellectual property.

Commercial firm licenses specify that they do not confer any right with regard to the logo and trademarks of the firm:

- Netscape public license: III. “Netscape and logo. This License does not grant any rights to use the trademark "Netscape", the "Netscape N and horizon" logo or the Netscape lighthouse logo, even if such marks are included in the Original Code”.
- IBM public License Version 1.0: “Except as expressly stated in Sections 2(a) and 2(b) above, a Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppels or otherwise. All rights in the Program not expressly granted under this Agreement are reserved”.
- APPLE public license: 10. “Trademarks. This License does not grant any rights to use the trademarks or trade names "APPLE", "APPLE Computer", "Mac OS X", "Mac OS X Server", "QuickTime", "QuickTime Streaming Server" or any other trademarks or trade names belonging to APPLE (collectively

"APPLE Marks") or to any trademark or trade name belonging to any Contributor. No APPLE Marks may be used to endorse or promote products derived from the Original Code other than as permitted by and in strict compliance at all times with Apple' s third party trademark usage guidelines which are posted at http://www.apple.com/legal/guidelinesfor3rdparties.html".

Choosing a different license for each situation.

Some companies have created a more neutral license for independent projects they have initiated themselves.

Thus, Netscape has two licenses for its free projects:

- the "Netscape Public license";
- the "Mozilla Public license".

Sun has followed suit with:

- the "Sun Community License Source";
- the "Public Sun License".

10.3.3 Copyleft licenses.

"Copyleft" licenses were created on the initiative of Richard M. Stallman: the General Public License (GNU GPL), and the limited GNU general public license (GNU LGPL).

These licenses offer the same freedoms as free licenses, namely, the freedom to copy, modify and distribute the Software[24] whether it has been modified or not. In addition, they organize access to the source code for the

[24] In this article, the term "Software"- with a capital S- indicates the software to which the license applies. The GPL not only applies to the "Software", but also to "the work that is based on the Software", i.e. the work which contains the Software or a version of this Software - with or without any modification - translated or not into another language (indeed, the GPL specifies that the translation is regarded as a modification).

Any element which can be identified and which is independent from the Software is not subjected to the license. Consequently, the license does not apply to such elements which are distributed on their own. But, when these same elements are distributed as part of an ensemble which represents a work that is based on the Software, the distribution must be done according to the GPL clauses.

In addition, putting together a work that is not based on the Software, with the Software (or any type of work derived from the Software), whether this is done through a storage or distribution procedure, does not automatically mean that this work will enter the GPL field of application. As long as an integration code is not observed, developments which are independent of the Software do not enter the license field of application.

As far as the LGPL is concerned, any program which is only conceived to function with the Library is not part of the license field of application. In theory, when all or part of the Library code is integrated into a software unit, this unit falls under the LGPL field of application.

However, the product which results from the combination of the Library with a program which is conceived to function with this Library, can escape the license field of application, while, at the same time, containing parts of the Library. On this assumption, if the LGPL does not apply to the product, it is necessary, nevertheless, that the license to which it is subjected, allow the product to be modified for personal use. It should also allow opposite engineering processes to be undertaken in order to correct mistakes if necessary. Moreover, certain specific conditions are required during the diffusion of this type of work:

- it should be clearly indicated that a library subjected to the LGPL was used for the design of this work,
- a copy of the license must be provided,
- the library copyrights as well as explanations on how to find a copy of the license must be printed next to the product copyrights if the latter are displayed,
- the entire library source code must be communicated,
- finally, it is necessary to respect one of the following points:

users and specify that it is allowed to invoice the physical act of transferring and copying an example. They also offer a warranty if the user is ready to pay for it.

The originality of these licenses lies primarily in the way these freedoms can be exercised. Indeed, when the GNU GPL license was written, the objective was to safeguard the freedoms of the user. However, this resulted in constraints that were sometimes too burdensome. That is why the GNU LGPL was envisaged: it does not have the same constraints but, consequently, it does not confer the same freedoms to the user either. This is why it is called: the "limited" general public license.

The conditions of the freedoms granted by the GNU GPL.

The GPL makes it possible for the users to copy and diffuse the software under the conditions that they:

- insert, in a perfectly visible and appropriate manner, on each example, a note concerning the copyright and a clause limiting the warranty;
- absolutely do not modify any of the notices which refer to the license and absence of warranty;
- give each of the Software consignee an example of the GPL along with the Software.

The GNU GPL makes it possible for the users to modify the Software and to diffuse the modified Software under condition that they:

- add all the modifications that were made in a very clear manner, as well as the date when these modifications were made. This obligation is justified by the need to inform the consignee that this is not the original, so that if a copy is faulty, it will not harm the reputation of the author of the original Software,
- distribute, under the terms of the License, the entire product containing either all or part of the program - with or without modification. It is thus allowed to modify the software under condition that it offer the modified Software the same rights as those mentioned in the GPL (and which, moreover, have made it possible to undertake the modifications),
- if the modified program reads interactive instructions normally while it is being executed, the user must make sure that, when he/she begins executing the program for such an interactive use, this program prints or shows a clear and simple notice which includes:
  - an appropriate statement concerning copyrights,

1. The entire library source code must accompany the work. When dealing with an executable form of product which is linked to the library, it is necessary to propose the entire product either as a source code or object code so that the user can modify the library.
2. The potential user must get a written offer which should be valid for at least 3 years. This offer should propose, in exchange of a fee that should not exceed that of the copy, either the entire library source code, or the supply of the product under the form of a source code when the product is executable through the library.
3. The sources must be readable, in other words, they must come with the appropriate data and tools (other than those which can be found in the distribution of the operating system) so that the user can generate the executable program.
4. If an access system allows distribution to a copy which is located at a specifi c place, then, at the same place, one should fi nd the same elements as those described here above (particularly, the version that comes under the library source code),
5. It would be wise to check that the users have received the source codes or that, at least, they have been told they will receive them, to conform with everything that has been mentioned above.

– and an appropriate statement about the fact that there are no warranty (unless provided by the user him/herself). Moreover, this statement must indicate that the users can redistribute the program according to the GPL clauses and it must also indicate how they can visualize an GPL copy.

None of this is necessary if the program is, by nature, interactive and if it does not usually print such notices on its own. If this is so, the work that is based on the program does not need to print such notices either.

In order to allow these modifications to be made, the access to the source code is guaranteed.

The GNU GPL allows the diffusion of the modified Software or not, under the object code or under an executable form, on condition that:

- the complete, corresponding source code be provided and be readable by a computer system. The code must be distributed through a medium which is usually used for software program exchanges,
- a written proposition should be provided. This proposition should be valid for at least three years and should mention that, anyone who should ask for the source code will be given a copy of it, readable by computer, through a medium which is usually used for software program exchanges. Its price will not exceed that of a copy,
- the information that has been received should be provided in order to supply the corresponding source code (this alternative is allowed for non-commercial distribution only and only if the concessionary has received the program under its object form or its executable form, according to the methods described in subparagraph b) above).

Several of the GNU GPL clauses aim at preventing any form of appropriation of the code:

- the user, whether he/she has modified the software or not, is not allowed to restrict the rights granted by the GPL to the people to whom the copies of the software are transmitted;
- it is not allowed to incorporate the Software or part of the software to other free software programs which do not have similar conditions of distribution without the author's agreement;
- finally, in an implicit way in version 2 and explicit way in version 1, the license prohibits the user from incorporating the Software or part of the software with other non-free software.

Conditions under which freedoms granted by the GNU LGPL are exercised.

The GNU LGPL makes it possible to copy and diffuse the Bookstore under the condition that:

- copyrights be inserted on each example, as well as a clause limiting the warranty;
- none of the notices referring to the license and absence of warranty be removed;
- an example of the LGPL be transmitted to the consignee at the same time as the bookstore.

The GNU LGPL makes it possible to modify and diffuse the Bookstore modified version on condition that:

- the user observe the same conditions as when he/she diffuses a non-modified copy of the library,
- the derived work be a software library itself,

- when the modified library uses information from the application program, without the information being parameters of the functionality, the result remains unchanged in the absence of this information.

Contrary to the GPL, the LGPL makes it possible to incorporate functionalities of the Library in another library subjected to a license other than the LGPL license. It also makes it possible to distribute the resulting library on condition that it include:

- the entire Library which is subjected to the LGPL,
- a clear indication that a portion of the library comes from a Library subjected to the LGPL as well as the place where one can find the non-integrated version of this Library.

Finally, the GNU LGPL makes it possible to diffuse the Bookstore, whether it has been modified or not, under object code or under an executable form, on condition that:

- the complete, corresponding source code be provided and be readable by a computer system, through a medium usually used for data exchanges. According to the license, the source code indicates the form under which the modifications can be most easily read.
- when the distribution of the object code consists in offering an access making it possible to copy the library from a specific place, equivalent access should be provided to obtain the source code from the same place.

License compatibility.

The principle.
By limiting the cases where it is allowed to incorporate the Software with other software subject to licenses which do not offer the same degree of protection with regard to freedoms of use, the GNU GPL becomes incompatible with all these licenses. Indeed, no physical integration of the codes is then allowed.

Consequently and before proceeding to any kind of incorporation, it would be wise to check if this is authorized. The GNU GPL invites the user to contact the author of the Software. In addition, the Free Software Foundation has made a list of the licenses which it considers to be compatible with the GNU GPL and those which are not. This validation (cf supra 1.6) helps the user know if it is possible to incorporate the software for all GNU projects which are related to software for which the FSF has intellectual property rights. As for other software, the fact that the FSF has validated such or such a license is a good indicator that lets the user know if he/she has the right to incorporate the Software.

Unfortunately, taking into account the number of licenses which are known as free licenses, it is not unusual to have to incorporate a Software program with another which is subject to a license which is not on the list. In this case, the only thing to do is to ask the authors of the Software if it is possible to incorporate the program or not.

Palliatives to the problem of incompatibility. It so happens that the authors of a software program modify their license or give it up for the benefit of the GPL so that their program can be incorporated with all the programs subject to the GPL.

Another practice has developed which consists in subjecting the same software to two different licenses, one of which is a GPL license. In this case, it is up to the user to choose between the two. Indeed, mixing different clauses is not allowed. This solution enables the authors to keep their license while allowing incorporations with the software that are subject to the GPL.

As far as libraries are concerned, the FSF has planned on using the GNU LGPL license which is less constraining than the GNU GPL license. One must keep in mind that each time it is necessary to resort to a library, it is also necessary to incorporate a program. Indeed, the GNU LGPL allows the incorporation of the Library functionalities to another library subject to a license which differs from the LGPL license and to distribute the resulting library (cf supra 3.3.2).

Patents concerning a computer program: the extent of protection.

FRANCK MACREZ,
University of Montpellier I/ERCIM, Franck.Macrez@free.fr.

When considered as an object of law, Software finds itself at the edge of the three following elementary categories: intellectual property, copyright (language property) and patents (industrial creation).

Patents are instruments of information dissemination and, as such, are conceived in order to promote innovation. It allows the patentee to benefit from a monopoly of the use of the contents that come under the definition of the term "patent". There is a counterpart to this monopoly, that is, communicating the way such an invention has been realized so that other individuals can understand how it works and eventually reproduce it (the patent office publishes such patents).

On February 20, 2002, the European Commission proposed a clause[1] which is working towards the legalization of the EPO[2] jurisprudence with regard to inventions implemented via a computer". This has sparked a controversial reaction among actors within the free software[3] community. In 1985, it was agreed that the software should be protected by copyright. Yet, at the same time, there were signs of increasing flexibility in patent law as a consequence of the famous Vicom[4] decision.

The question has been put back on the agenda during the past few years. One can explain the evolution of the jurisprudence of the EPO appeals courts by the influence of American and Japanese policies. Both countries grant patents for software and commercial methods. Another factor in this evolution may also come from the desire of software editors to better develop their products from a financial point of view by giving them an industrial title. Having a patent makes it possible to show the product in the balance sheet. This was particularly interesting in the context of the speculative bubble from 1999 to 2000.

Thus, there seems to be a misappropriation of patents through the practice of "façade patents", whereby the patenting process is, most of the time, obvious (and thus easily rescindable). These "façade patents" would make it possible for multinationals in particular, to put pressure on small software editors in order to free them from having to pay royalties (there has been talk about "small swindles" in this regard[5], some of which are bigger than others)[6].

In addition, opacity seems to be the rule in this field. The majority of these patents are not that easily understood, even for a specialist. Nevertheless, disputes are rare. For businessmen, secret compromise is the most rational solution. Indeed, legal procedures take so long that they are incompatible with the average lifespan of a specific software technology[7]. And even when a dispute does take place, the judge refers primarily to the unpublished expert report.

These "façade " or "barrage" titles are actually intended to be rescinded as soon as the first dispute is presented in court. However, is it fair to assert that these "software patents" only constitute an economic and financial strategic weapon? In spite of the difficulty this opacity generates for those seeking to understand what is happening behind the scenes, it seems fundamental to us to try to define what is truly protected and what is not. It is necessary to distinguish between what is rescindable and what is valid. The threat comes from the potentially durable legal insecurity that could result from this situation. People had been waiting for years for the clause suggested by the Commission (yet, its contribution compared to the already pre-existing EPO jurisprudence is insignificant). An agreement with the European Parliament for the effective vote of a clause should prove difficult. Moreover, if one takes into account the European Community procedures, such an agreement is unlikely for several more years[8].

[1] http://www.europa.eu.int/comm/internal_market/en/indprop/comp/com02-92fr.pdffor the French version.

[2] The European Patent Office.

[3] Principally Aful, April, Eurolinux, FFII.

[4] Chamber of Technical Recourse (CTR) of the European Patent Office (EPO), July 15, 1986, VICOM System Inc, PIBD 1987, 409-III, p. 134.

[5] Dominique FORAY, meets APP, 28/05/02.

[6] For the description of a concrete situation, http://swpat.ffii.org/papiers/eubsa-swpat0202/index.fr.html.

[7] *ibid.* "(...) You are now a successful company. A patent agency approaches you: you are in violation of 2 or 3 of their patents. The field in question is broad. They want €100,000. A legal battle could block to you for 10 years and to cost up to a €1 million. You end up paying. One month later another patent agent knocks on your door..."

[8] Although we are optimistic, we can estimate that the "shuttle" procedure will last around 2 years; it is necessary to add a time

The extent of protection of a patent concerning a computer program can be determined upon its creation (I) - the delivery of the title by a patent office - and at the time of the implementation of the patentee's prerogatives (II). It seems significant to us to distinguish between the two different realities which come under the phrase "invention involving a computer program": the computer program is only one innovation at the heart of an overall process in which many other innovative elements intervene.

11.1 The birth of a patent.

A patent application, defining the object of the propriety, through descriptions and claims, must relate to a "technical" creation.

11.1.1 Descriptions/claims...

The claims define the extent of the protection sought for the invention, but they are interpreted according to the descriptions which accompany it.

Editing the descriptions.

A specific language.

There is a specific technique concerning how to write a patent. If this is not respected, the patent office will not deliver it. In order to avoid phrases like "computer program as such" which often results in the exclusion of the project (Cf. *infra*), it is primordial to materialize the "invention" by stressing the physical aspects of the function of the software: its action with regard physical connections for each peripheral("Entry/Exist signal"), even with regard to components (memories, hard disk, diskette read-write head, for example).

In order to register a patent concerning a software program, it is necessary to "use terms such as "process", "invention", "system" or "device", rather than "software" or "programs"[9]. It is wise to choose the phrase "interactive handling system" rather than an "information processing system" or "means of initialization" rather than "resetting the register back to zero", "compression device" for an algorithm having the same function[10] or "reservation of several memory blocks for a variable in a memory device".

Publishing sources.

This should be done in a natural language, eventually accompanied by explanatory diagrams. The description must be adapted to the nature of the invention: "At the end of the description, in the appendix: 1°short extracts of computer programs, presented in the form of print-outs written in regular programming language (...) may be added"[11].

The source code itself is not sufficient to comprehend a process. The flow chart - as a way to complement the descriptions in a natural language - appears to be the most adapted means for *the description/comprehension* of a particular use of an algorithm.

In reality, very few people reveal the source code of the program concerned. This is unfortunate with regard to the reproduction of the invention as well as to the efficiency of the background check (and, of course, with regard to the dissemination of the information since this is the original objective of the patent system).

limit of about 3 to 4 years for the transposition of the directive.

[9] Guide Lamy [2000], n°3415.

[10] Examples quoted by Linant de Bellefonds [2001], p. 9.

[11] Article R.612-13 CPI.

When somebody claims a product (Cf. *infra*), its description should initially permit *the reproduction* of the "invention"[12]. A law recognizing the validity of the software patent (organizing the patentability of the software) should make the disclosure of the source code compulsory whenever something is claimed to be a product.

The form of the claims.

From the process to the product.

In "software patents", the claim will always be functional. However, the claim can relate to a product[13] or a process[14].

Since 2000, both the INPI[15] and the EPO[16] have accepted product claims. Considering that software is a process of information processing, such a decision may sound surprising at first. This choice can be explained by the patentees' "need" to deal with procedures of seizure more easily. It takes much longer to prove the use of a process (within the buildings of an seized company) than to seize a machine directly, even only a hard disk, or any other support.

This is why patents relating to a product-program[17] claimed as a hardware support, have gotten positive reactions. They usually read as follows: "computer product-program which includes program code portions/means/instructions recorded on a hardware support that can be used in a computer (...)"[18].

But the Commission has gone back on its decision and now considers that this would amount to patenting programs "as such"[19]: only a "product-machine" can be claimed[20].

How to determine claims.

Software deals with processing information. It can be considered as part of a broader industrial process[21]. The method which has been adopted by the EPO to determine the validity of an invention as a whole[22] — contrary to the method which consists in sticking to the invention's "essential core"[23]— makes it possible to consider both the software and the hardware together, whereas the single new and inventive element can be the software alone. The software is considered "as such"...[24]

Along the same lines, Article 5 of the clause that was suggested by the Commission wants "the Member States to make certain that an invention implemented by computer can be claimed to be a product, that is to

[12] Vivant [2002]

[13] "A given body defi ned by its mechanical composition or its chemical structure": "... a device, a machine and, more generally, any other object thus constituting a product" *Lamy Droit Commercial*, n° 1577.

[14] *Ibid.* "As for the process, it consists in the implementation of a certain number of physical means, chemical or mechanical, whose structure makes it possible to obtain a certain result, consisting either of a product, or an immaterial effect "

[15] Examination instructions concerning inventions related to computer programs and/or commercial methods, 00/10/26, unpublished.

[16] Decision CRT 1194/97—3.5.2, OJ EPO, 12 / 2000, p. 525.

[17] The expression means a claim of the hardware-software couple, of a hardware support or machine on which the program is downloaded. The meaning is thus not the same as the terminology employed by a progammer, where the "product program" would gather: documentation (customer and technique), the program (algorithms and data structures), and tests (checking and validation).

[18] Examination instructions of the INPI, op. cit.

[19] "Proposition for a Directive", op. cit., article 5, p. 16.

[20] Cf. "La théorie d'origine hollandaise de la "machine virtuelle": Programming a machine in various ways makes it possible to consider that one is in the presence of different types of machines.

[21] "Affaire Schlumberger", Paris Court of Appeals, June 15, 1981, PIBD, 1981, III, p. 175.

[22] CRT 3.4.1, May 21, 1987, T 26/86, "Radiological Equipment" / Koch and Sterzel, PIBD 1988, 432-III, p. 185; T110/90-3.5.1— April 15, 1993 "Document presented under a version likely to be published" / IBM, OJ EPO, 1994, p. 557.

[23] T22/85-3.5.1 - October 5, 1988 "Résumé et Recherche de documents" / IBM, OJ EPO 1990 p. 12; T38/86-3.5.1, February 14, 1989, " Traitement de texte" / IBM, OJ EPO 1990 p. 384.

[24] It is this interpretation which one can discuss in order to call into question the validity of the patent. Cf. *infra*, II.

say, as a programmed computer (...)". Programming a machine in different ways allows one to consider that one is dealing with different types of machines.

11.1.2 ... for an invention.

Whether the object of the patent derives from the concept of an invention[25] or whether it derives from industrial applications, it must have a "technical" dimension. The process which is claimed must offer a "technical contribution", which "is defined as a contribution in a technical field which is not obvious for the specialist in the field"[26]. Beyond the need to distinguish between what is technical and what is not, the main objective is to aim at an innovative technical function.

An innovation.

Two conditions are required: the invention must be new and it must not be obvious for "the specialist in the field". It is important to note that the concept of originality, which is a pre-condition that is necessary to apply copyrights and which is objectivized via an "intellectual contribution" in order to adapt copyrights to software, is very close to the novelty of patent law.

Novelty.

Once again, the question of the mode of description (Cf. *supra*) can re-appear on the agenda. If properly adapted, it allows for an effective background check by determining the state of the relevant technical application.

As far as patent offices are concerned - this is not specific to software patents - the background checks which are cited are, in the vast majority of cases, patents or review articles and not a software program whose source code has been opened[27] As a consequence, it is not that difficult to overcome the novelty condition which is required for the delivery of a title: as long as the background check shows that no such things already exist — that is to say, that no modification, however minor, has been made to "the technical form or function of the means, results, combinations or applications"[28] —this is enough to consider the process to be a novelty.

At any rate, if it is clearly proven that there is no trace of novelty, the judge will cancel the patent.

Inventive activity.

Inventiveness means a technical development which is not obvious for the "specialist in the field". Using a "problem-solution" approach, the EPO inspector must determine the objective technical problem that has been solved thanks to a process which encourages technological "advancement".

The contribution will lie in the means which are claimed (i.e. technical characteristics), in the effects obtained, or in the problem itself: the appreciation of such a contribution cannot be made if one already knows what the problem is, since the latter would be integrated fictitiously in the technique[29]. What matters most is to "properly formulate the problem, that is to say, ask the right questions instead of [ ] solving this problem, which any specialist can do once the question has been properly formulated"[30]. Indeed, the CRT of the EPO points out that: "discovering a problem that is not well-known can constitute, in certain cases, a patentable

[25] Classic definition: "a technical solution to a technical problem thanks to technical means which are likely to be repeated", Mousseron, Traité des brevets, Librairies techniques, 1984, n°154, p. 175.

[26] Proposition for a Directive, op. cit. p. 14.

[27] For a statistical study of this practice in the American Patent Office, Cf. Greg Aharonian [2000].

[28] Pollaud-Dullian [1999].

[29] "It is the statement of the problem which was not obvious. It is in this statement that the inventive step resides"; Mathély, "Le nouveau droit français des brevets d'invention", Mathély [1991], p. 95.

[30] Smets, above mentioned note 24, p. 44.

object, even if retrospectively, the solution that is been claimed seems banal and obvious"[31]. Nonetheless, this problem is likely to be stated in the terms of reference by the client of the person who is applying for a patent.

As for the judge, he/she will look for certain criteria in order to determine whether or not the patented invention is genuinely inventive: in other words, he/she will consider the gestation period of the invention, whether a difficulty or a prejudice has been overcome or whether a decisive break with the current technique has been made.

A technical function.

We will only deal with the technical function of information processing that the program carries out and for which an invention will always be a technical solution to a technical problem. However, for example, if it were related to the physical state of a device, the processed data would confer a technical aspect to the entire process[32].

A technical problem.

As we have seen above, "what matters most is to properly formulate the problem". The criterion concerning the "technical considerations" released by the EPO[33] shows the limitations of the concept of "technicity". There will be a technical contribution —and thus the delivery of a patent— if one can find a "whole set of activities utilizing technical considerations".

An example of a technical problem (whereby the processed data is of technical nature) is: "the visualization of information about events which occur in the input/output device of a word processing system[34].

A technical solution.

The function of the program, which should not again run the risk of being excluded from article 52 CBE, will be to produce a "technical" effect.

Whatever the program, a computer produces electric pulses. To be patentable, the technical effect must be "supplementary", that is to say, it must "go beyond normal, physical interactions"[35] that occur between a program and a computer. "When a program constitutes a modification of data and does not produce any other effect than mere information processing, it is excluded from the field of patentability"[36]. The technical effect could be produced by a functionality which would use the resources of the machine in a powerful way. However, if it is only the software that generates this additional effect, the applicant (or patentee) will be denied the so-called "computer program" exclusion. It seems, in fact, that the "supplementary technical effect" will result from a new peripheral, hardware which will be patented at the same time as a software component.

[31] T 2/83, March 15, 1984, OJ EPO 1984 p. 265

[32] "The automatic viewing of information concerning the state of a device or a system is primarily a technical problem"; Civil Court of Appeals, EPO September 5, 1988, "the method for decoding expressions and reading events in a word processor "/IBM, PIBD, 476-III, p. 255. " the printer command codes must be regarded as design features of the word processor "; T 110/90, Civil Court of Appeals 3.5.1, "document presented under a version which is likely to be published / IBM", OJ EPO 1994 p. 557. s. also (and already) the Vicom decision (mentioned above) in which the CRT noted that the method that was claimed remained abstract as long as "the fact that data represent a physical entity was not specifi ed in the method"...

[33] T769/92-3.5.1, Sohei / "universal management system " - OJ EPO 1995 p. 34. The request related to "a computer system for various types of independent management which include, at the very least, fi nancial and stock management functions..." and whose strong point lies in that it allows the use of a "single transfer slip" for all the management operations that are targeted.

[34] T 0115/85, September 5, 1988, OJ EPO 1990 p. 30.

[35] CRT Decision, July 1, 1998, "Resynchronisation asynchrone d'une procédure de validation", T 1173/97-3.5.1, JOEPO, 10/1999, p. 620.

[36] CR EPO, Siemens, December 12, 1989 T158/88.

The Commission reassures us by stating that "an algorithm[37], defined without any reference to a physical environment, is not technical and, consequently, cannot constitute a patentable invention"[38]. It is true that other exclusions mentioned in Article 52 CBE relate to "mathematical formulas", "presentations of information" and "scientific principles". The object of the patent could be narrowed to the relation between the abstract principle and the *precise* function which is claimed, namely, the specific implementation of an algorithm. In this case, it is the description of an algorithmic, a science which studies the application of algorithms to information technology, which would be patented.

The ambiguity inherent in the "technical effects" doctrine makes it impossible to specify the focus of these patents concerning computer programs with any degree of certainty. How does one define what is *technical?* Each activity has —in a nonexclusive way— its own technological applications. This word can stand for almost anything[39]. One could distinguish between: individual technological applications, social technical applications, intellectual technical applications and real technical applications. These are just a few technical applications among others which can be patentable: technical applications "which aim at modifying the immediate external world whether it falls within the domain of what is considered organic or inorganic"[40].

Behind this "technical application" requirement, the question remains the same: what do we mean by the term "invention" whether it concerns the moment when a genetic sequence is isolated or to the implementation of a method by computer. The debate on "scientific property"[41] does not explicitly re-appear and yet, this is the main question: what do we *want* to consider as an "invention"?

11.2 Life of the patent.

Determining the sphere of reservation is accomplished by examining the particularities of the invention patent with regard to counterfeiting as well as its relation to copyrights.

11.2.1 Counterfeiting a computer program patent.

Let us push aside from the start the particular case of partial counterfeiting[42] (whose existence is recognized by the Supreme Court of Appeals and, in most cases, rejected in doctrine), which has no reason to exist in this matter. The individual who has patented the following combination A+B+C whereby A is a program, will not be able to claim the protection of A alone since this would mean seeking the protection of the program as such.

Along with the classically prohibited acts (1), it will be necessary to adapt the theory of equivalents (2) which is traditionally found in patent law.

[37] "Operating rules suitable for calculation or data processing. —Calculation, sequence of actions which are necessary to achieve a task" Le Petit Robert, 2001.

[38] Above-mentioned proposal for a directive, considering n°13, p. 20

[39] Weber [1971], p. 63:... one can talk about a technique of prayer, a technique of asceticism, a technique of reflection and research, a technique of mnemotechnics, a teaching technique, a technique of political and hierocratic domination, a technique of war, a musical technique (that of a virtuoso, for example), the technique of a sculptor or of a painter, a legal technique, etc. Besides, they all are subject to an extremely variable degree of rationality". Following a similar list, Dr. Swen Kiesewetter-Köbinger adds, with a touch of humor: "Even with reference to the sexual act one speaks occasionally of special techniques", *On the Patent Examination of Programs for Computers*, http://swpat.ffii.org/treffen/2001/bundestag/kiesew/index.de.html.

[40] Von Gottl-Ottlilienfeld [1914], quoted by Goffi [1996], p. 23.

[41] "How can one accept that it is possible to obtain an honorific title for a mere industrial application, whereas the discovery of essential scientific laws, which perhaps control a whole series of applications, would not bring any rights at all to the scientist? The latter gives, for free, the object of his/her work, yet, this is the most meritorious aspect of his/her work. Society seems to emphasize the prosaic work of the inventor only because he/she deals, in a more immediate way, with the satisfaction of human needs." Roubier [1952], p. 55 and many other quoted references (this debate is an old one). See also de Laat [2000] pp. 187-204.

[42] This consists in protecting one single element of an invention which has been claimed as an ensemble.

Prohibited acts.

The patentee's prerogatives.

It is logical that when an individual claims a product to be their own, the patent allows them to prohibit its manufacture, use and sale (Article L 613-3 of the CPI)[43]. This concept must be understood in a very broad sense: "selling a product must be defined as any type of material operation tending to put a product into circulation, regardless of the quality of the conditions in which the author has carried out this operation"[44].

When it is a process which is claimed, all use is prohibited without the patentee's authorization. The same prohibition applies to the product resulting from the process.

But there are exceptions to the patentee's rights: Article L 613-5: "the rights conferred by the patent do not extend to:

1. acts that have been carried out within a private framework and with noncommercial ends in view;
2. acts that have been carried out on an experimental basis which relate to the object of the patented invention".

In order for a judge to prohibit such acts, it is necessary for the defendant to prevent the canceling of the patent in question. One can simply note that, considering the degree of technicality of question here, the judge actually has "a quasi discretionary capacity of decision (in optimistic terms, one can call this power of equity)"[45]...

The canceling of the patent by the defendant.

The first argument for canceling a patent that the defendant will have to consider is the lack of novelty and inventive activity (Cf. *supra*). As we saw earlier, the EPO interprets the claims in their entirety in order to determine whether or not they have "technical applications". It seems to us that the defendant can try to prove that the only innovative aspect of the ensemble is a computer program. Consequently, the patentee, the plaintiff, seeks the protection of a program as such.

Strictly speaking, prohibited acts do not only relate to the use of a process or product. Any equivalent violation will also be regarded as counterfeit.

The indefinable equivalent of "software means".

The definition of an equivalent technical function.

The theory of equivalents makes it possible to prohibit the use of *equivalent* means to patentees, that is to say, a means which is different but which fulfills the same function so as to obtain an identical result. In patent law, the function is defined as the ensemble of the "primary effects" realized through the execution of the process or product.

[43] "What follows is prohibited, in the absence of the patent owner's consent:

1. manufacturing, offer, introduction on the market, use, importation or detention, for the purposes mentioned above, of the product which is the object of the patent;
2. the use of a process which is the object of the patent or, when the third party knows, or when circumstances make it obvious that the use of the process is prohibited without the patent owner's consent, the offer of its use on French territory;
3. the offer, introduction on the market, use, importation or detention, for the reasons mentioned above, of the product directly obtained by the process which is the object of the patent."

[44] Paris, Dec. 3, 1985: PIBD 1986, 388, III, 130.

[45] Foyer and Vivant [1991], p. 168

Thus, the equivalent will be that which allows one to obtain the same technical effect thanks to a method which has a different form or structure. It is clear that the field which is claimed can be very broad indeed[46].

Patenting a result.

The means would be a method, the description or the implementation of an algorithm. Already the "distinction made between the result and the technical function the means [was] riddled with problems"[47] when mechanical inventions were dealt with. Protecting the technical function of a computer program by a patent would amount to allowing the result to become the propriety of the patentee. Patenting a result is traditionally prohibited under French law.

In the software field, one must rely on the judge's wisdom. He/she will nonsuit the patentee putting forward the reason that equivalent means were used. However, it seems to us that having the choice to reject a whole aspect of the traditional doctrine of patent law should be the responsibility of the legislator rather than that of the judge.

This is how we encounter some absurd situations. For example, when a patent is granted for an invention which includes both the hardware and software, "substituting new software for software which would have made it possible to obtain a patent could be regarded as infringement under the doctrine of equivalence. To provide such software by marketing it and making it available to the general public could become an act of infringement through "supplying of means"[48]. The software which activates a given hardware (one thinks of the peripheral pilot) will not be reproduced so that this hardware works for another computer system, for example. This obviously makes things more complicated when one studies the copyright regulations, particularly when it comes to guaranteeing possible interoperability between different computer systems.

11.2.2 Towards a combination of protection.

The patent would protect the subjacent principles of a program. The written part of the program —its layout— would be protected by copyrights. Implementing such a principle is far from easy. It seems impossible to put the concept of coexistence into practice, that is to say, defining two different objects of law. This is why combining different protection systems seems to be the only solution[49].

Copyrights and protection of ideas.

As far as copyrights are concerned, the software is compared to a literary work. Its form is protected and this includes the structure of the work.

Protecting the form only.

The principle of the copyright is to protect the way a work is expressed, and not the underlying idea. As for the software, the protection obviously applies to the source code, but also to the object code, although its form is volatile and depends, in particular, on the material architecture on which the source code has been compiled.

[46] British Telecom was thus trying to have its patent implemented. The patent supposedly deals with hypertext links by interpreting the text of this patent through equivalents: "blocks of information " would apply to any html file, a "complete address" would represent any Internet URL.

[47] Roubier [1954], p. 82.

[48] Vivant and al. [2001], p. 109.

[49] "Protection can be cumulative in the sense that an act which implies the exploitation of a specific program can undermine copyrights which protect the program code. The same is true for a patent which protects the ideas and principles which are subjacent to this program"; the directive proposal mentioned here above, p. 9.

Of course this form must be original. This means that the programmer must have had choices in the writing of the program: he/she must have made a "personalized intellectual effort". The criteria that the judges accept to characterize the originality of a software program can be quite similar to those used to evaluate the inventive activity of a given invention (Cf. *supra*).

Protecting the structure.

The exclusion of the ideas of protection by literary and artistic property is a principle, but —once again—, it is difficult to draw the boundaries: "It is impossible to make a precise distinction which would show where punishable counterfeiting stops and tolerated plagiarism begins"[50].

The criterion of composition, as formulated by Desbois, makes it possible to protect "the development of general ideas"[51], for example, plans, work organizations. In applying this criterion to software, it is the internal structure, that is to say, the architecture of the program, that is being considered.

A complex articulation.

There is only one example which combines different kinds of protection between an industrial title and a copyright, namely that of industrial design and models. Moreover, defining the "inseparable or separable aspects of the creation of form and technical creation is an extremely delicate operation" in this field[52]. This distinction is even trickier in the software field.

Antagonisms between copyright and patent.

It must be acknowledged that the patent confers greater freedom to third parties than copyrights do. The latter allow one backup copy only (outside of a "private framework") and close the decompilation ("experimentally") under strict conditions (when the objective is that of systems interoperability).

Moreover, when applying for a patent, the description of an algorithm could take on the same written structure as that of the program itself (i.e. an algorithm is a structure). In the case of a program written in a weakly marked computer language where "structure" versus "writing" appropriateness exists, retaining the protection of the structure through copyrights would mean recognizing that the published text of the patent would be counterfeit. This would also mean that the protection of the algorithm would be guaranteed by copyrights as well!

Would combining different kinds of protection offer greater freedom?

Of course, specific legal clauses would be necessary to organize such a combination of different kinds of protection. The clauses could read as follows: "the clauses concerning copyrights do not undermine those concerning patents, and vice versa".

The first part of this proposal is stated by the Commission: "exercising a patent which applies to an invention implemented by a computer must not interfere with the exceptions which have been granted by virtue of the legislation covering everything from copyrights to software designers by the clauses of the 91/250/CEE directive[53]. In other words, decompilation will always be possible.

What about the reverse? It would be necessary to add that copyright clauses do not undermine exceptions to patent law either (particularly the exception made for private use with noncommercial ends in view). But this would cancel the copyright legislation which is specific to the software: the limitation to one backup copy

[50] Pouillet, quoted by Lucas, Traité de Droit de la Propriété Intellectuelle, p. 240.
[51] Quoted by Lucas, *ibid.*
[52] Bertrand-Doulat [1997], p. 63
[53] Directive proposal mentioned above, p. 9.

(more restrictive than the "traditional" private copy) would then have no purpose with regard to any patented program!

Conclusion.

For the time being, the legal inconsistencies pertaining to software protection are quite obvious. This is true for both the principles as well as their implementation. It is clear that patent law needs to be adapted in a substantial manner if one wants to insert the computer program in the sphere of reservation of industrial property. If, indeed, there is genuine political good-will to implement these adaptations (which the law is simply laying out), it seems to us that the "patent concerning a computer program" will no longer be a real patent as such, but a mere industrial title (just like copyrights which are specific to software, in fact, resemble *sui generis* law).

It remains that, as we have seen above, it is the combination of both hardware and software which is currently patented in a *valid* way, the hardware being the new and inventive element.

Conclusion: Towards a New Software Economy?

JEAN-MICHEL DALLE,
University of Paris 6 & IMRI-University of Paris Dauphine, Jean-Michel.Dalle@admp6.jussieu.fr.

12.1 Introduction: not seeing the forest for the trees.

Where does the “new software economy” really stand at the present time? It is our belief that on-going changes will once again revolutionize the software industry, an opinion shared by the vast majority of IT actors[1]. This interpretation emerged primarily as a consequence of the arrival of free and open software programs during the 90s. They began filling new slots in the market, imposing their economic models and acquiring market shares. They even began to threaten their competitors, the producers of proprietary software programs. Their emergence also brought strong hopes that, "finally", a viable production model based on software components, or COTS ("components-off-the-shelf"), would come into existence.

First of all, we must admit that the economic context around the year 2000 was significant. More recently, however, this "new economy" has had its ups and downs. We have witnessed the financial rise and fall of dot-com companies and, more generally, start-ups. What was the importance of this context on free software: in other words, was there "path-dependence" (David [1985])? How much of this recent history is responsible for the current situation and what is to become of the new software economy? Right now, this question remains largely insoluble, but it is clear that the nature of available financing weighs heavily on the choices companies have to make with regard to the selection of economic models. Indeed, an essential dimension of an economic model is the type and duration of the investment which it requires. Economic models which have exploited quick market shares - and thus standardization strategies in order to become the dominant standard, models which therefore brought in considerable investments over long periods of time - have not frightened investors. They permitted the initial rise of "distributors", primarily Linux (Red Hat, Mandrakesoft, Suse, Caldera, etc). These investments obviously corresponded to particular financial market conditions which no longer exist at the present time. For all that, the case of Linux distributors is highly specific. It is characterized by standardization strategies that cannot be found in other free software programs. As a consequence, it may paradoxically have been the "tree" which stopped other companies associated with the development of free software from "seeing the forest" and, more generally, from understanding the evolution which now characterizes the new software economy. This is what we shall try to clarify in this conclusion. Admittedly, these companies have benefited, to a certain degree, from buoyant financial markets, robust private funds

[1] At the time when this project was launched, its main objectives were to favor collective training around these new and complex questions. It was also meant to help a community of thinkers and researchers to get together via workshops which allowed experts and academics to meet and discuss. This was also made possible thanks to the work of young researchers who were interested in the subject. I want to thank Nicolas Jullien for having agreed to coordinate the final months of this project, a period when I was obliged to devote my time to other tasks. It is mainly to him that we owe the final report. I want to thank him again as well as Mélanie Clément-Fontaine, without whom this project, would never have seen the light. I also want to thank Laurent Kott for his support and interest in these questions. He played a very significant role and helped us move forward. Thanks are also in order for Marie Coris, Olivier Dupouêt, François Horn, Franck Macrez, Laure Muselli and Aymeric Poulain-Maubant for their excellent and relevant contributions. Finally, I want to thank all the participants and speakers of both workshops organized at the INRIA, in particular, Christian Genthon, Michel Bidoit, Gérard Roucairol and Hughes Rougier. More personally, innumerable discussions helped me better understand the free software economy, and I simply cannot mention all those who have thus contributed to this project, either by bringing new ideas/functionalities, or by "debugging" some of my ideas. However, I especially remember my weekly discussions with Michel Bidoit, Gérard Giraudon, Hugues Rougier and Guillaume Rousseau, particularly during the drafting of the software group report for the French “Planning Commission” (au Plan [2002]). In any event, if there remain any bugs - they are mine - and new ideas still need to be forthcoming: any form of contribution is obviously more than welcome!

Mention should also be made of the spirit which animated this project. Indeed, it seems to us that one of our successes was to prove that economists and lawyers can collaborate very well on the "open-source" question and that they do not necessarily reflect the opinions of larger lobbies. Finally, it shows that they can often exchange ideas without being constrained by any form of allegiance. In a sense, the question of free software is an open question in itself which economists have defended in order to contribute to finding the best solution in terms of social well-being, beyond more academic types of work. Indeed, even if motivations and incentives are very significant, it seems to us that questions concerning a new software economy (and open software in general) may have been somewhat neglected. This is undoubtedly changing quickly, in particular in Europe. Indeed, we were given many opportunities to present the work we have been doing on these topics outside of France: in 2002 alone, in Athens, Brussels, Copenhagen, Madrid - and also Washington DC.

investments and, in particular, venture capital funds.

Conversely, however, one can also interpret this phenomenon as something that has allowed for a much faster exploration of economic models than one might normally have expected. Namely, it allowed for much more rapid progression in the field of collective training[2]. This is a classic Schumpeterian approach, the originality of which lies in its application to an economic rather than a technological rupture. Just as any technological rupture is accompanied by a whole set of tests and errors, which often correspond to a specific group of companies, a socio-economic rupture usually is accompanied by a phase of exploration. Consequently, this exploration is strengthened by the fact that a high number of new firms have been created and that they are working on many different approaches[3].

Does this Schumpeterian model still apply when there is no interest in promoting a truly revolutionary technology? We believe that the underlying concept is still valid, but less markedly so, since the market dynamics involved are no longer the same. The point is not to sell a new product. It is to sell a product that one's competitors do not have. The companies' ability to dispute existing markets and to protect themselves thanks to sufficient entry gates varies considerably. In other words, the rupture may in fact miss its objective and not impose itself. For technologies which are characterized by network effects, the question is also one of a standardization economy, which is quite different from that of technological rupture. It deserves to be studied in such terms (for a recent review of standardization economy, cf. Foray [2002]).

Contrary to other new technologies, free software has not necessarily benefited from the comparative advantages a major technological breakthrough can confer. Free software programs have adopted strategies to penetrate markets, replace existing products and thus progressively propose standards by using powerful network effects. As far as Linux's distributors are concerned, it would be completely erroneous to reason in terms of economic models with regard to the firms involved, since the question deals, first and foremost, with standardization. When financing is easily accessible, one standardization strategy consists in trying to acquire as many market shares as possible and as quickly as possible. However, this strategy has proved rather unsuccessful and has now been abandoned. Other strategies have emerged which all have to do with standardization: Red Hat has shares on the NASDAQ stock market and MandrakeSoft has been listed on the Paris Open market Stock Exchange (this strategy provides both transparency and financing). Several distributors have also regrouped and Suse has proposed an interface between its distribution and Microsoft's Office pack. In the same way, certain MandrakeSoft distributors have turned towards niche strategies, such as, for example, education. They then tend to differentiate their products. But the difference with traditional editors remains very small indeed: the only thing that might be slightly different are the standardization strategies.

Thus, in this conclusion, it is important for us to highlight the phenomena which truly characterize the

[2] Above all, it is in terms of collective training and the training curve, regarding the question of open software programs, that it is necessary to interpret the contributions presented in this report, and, more generally, the RNTL NME/NEL project. This shows how prudent we must remain and how we must avoid giving "definitive" answers since they become, almost instantaneously, obsolete. Thus, in this project, we have striven to work like "scouts" and blaze some new trails which can shed some light on the matter. This has been a priority for us. We also lacked sufficient data at the time when this project was launched: the first figures with regards to the cost of free software (TCO) are thus recent and still relatively contradictory since, to date, we do not think that any type of survey has been done which really takes into account the costs of changes and training, and not simply those of acquisition. Even more significant, it is difficult to evaluate the extent to which free software programs have been adopted in terms of market shares. Indeed, except for some known and meaningful figures, this evaluation concerns the firms themselves, and this is only the tip of the iceberg. Not that long ago, it was not that easy to know which firm or, for that matter, which public service used Linux. It was just as hard to know who, within a particular "traditional" SSII, provided services around free software programs. It seems to us today that if this project were to continue, it should go in this direction. It should contribute to the clarification of these questions in a quantitative manner at a time when this market is truly starting to move forward.

[3] Quite simply, such a Schumpeterian vision has also animated a certain number of actors and observers of the "new economy".

new software economy as well as the "innovations" it is likely to produce. First of all, we insist on the fact that free software economy is mainly an economy of bundles and, more precisely, an economy of layers. In a sense, this is true for any type of software economy. However, the fact that it is free makes this phenomenon even more relevant. It explains the emergence of free software service companies (SS2L) as well as those of promoters, but also the recent emergence of "specialized integrators" and "quasi-editors", as we call them. The latter are especially linked to the fundamental nature of the new software economy, which is to promote a component model. Langlois [2001] describes this as a true "modular revolution". More exactly, it is about an economy where system integrators (Pavitt [2002]) have to play a significant role and where the opening of the sources ensures the presence of easily adaptable "off-the-shelf" software components.

12.2 A "bundle" economy.

The economy that is associated with free software is primarily that of "bundles". Since free software programs cannot be sold directly, without any regulations, they are almost always free. The overall objective is to sell something else with which they are "associated" and which, ideally, comes with or at least is useful to such programs. A typical example of a bundle economy is that of razor blades: the razor could almost be given away for free, as long as people keep on buying blades. In the field of free software the logic is similar. Thus, packaging, documentation, on-line assistance and other similar services can prove to be useful for free software users. As a consequence, they are likely to be sold in a "bundle" along with these software programs. However, packaging and on-line assistance are not particularly useful when dealing with well-trained developers. This may explain why some free software distributors have had some difficult times. On the other hand, companies like *O' Reilly*, which is specialized in the edition of books and handbooks dedicated to free software programs, have been incredibly successful. Yet, this is another bundle economy model which, in this case, provides documentation. In all cases, it is the nature of the request which conditions the nature of the bundles that are likely to be proposed on markets which, at times, are segmented.

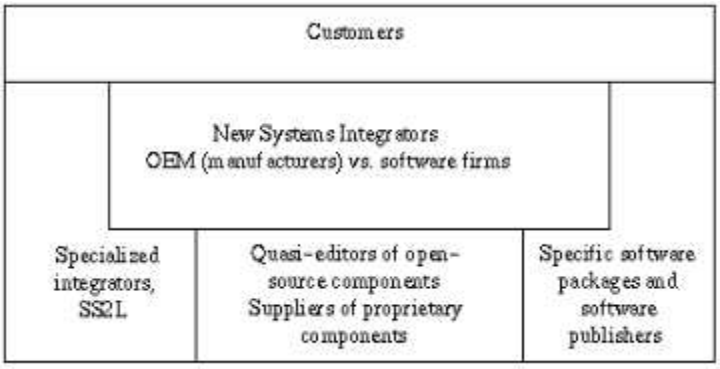
As a whole, associating two kinds of goods can add extra value to each of them when they are taken separately. Economic actors can use some of this added value. With regard to software programs, one of the most natural bundles on the professional market consists in associating software programs and services. If a customer does not have the skills required to install a software program, this is not always easy to accomplish. Moreover, the customer often finds it necessary, and almost always useful, to have a program that is adapted to his/her specific needs, although this is in the very nature of software goods. It is this simple reasoning which explains why SSIIs have been so successful. This has also been the case for free "SS2L" (Free Software Service Companies) software. However, in the case of free software, recruitment is particularly essential because it has to do with developers who are usually independent and come from developing communities that are associated with the free software industry. Thus, SS2Ls generally make some effort with respect to the communities of developers (cf. Coris [2002]). They allow them time to contribute to free software projects and they themselves initiate such projects. They also hire well-known developers and allow them to develop whatever they want. They respect the specific culture of programmers who come from such communities. They even adopt specific organizational modes, which are very similar to co-operative forms, in order to facilitate their recruitment and develop a culture which allows them to establish employee loyalty and develop new bonds. Such practices should also be interpreted in signaling terms with respect to the labor market for developers specialized in free software.

The SS2L economic model is one of the most frequently mentioned models when people talk about the free software industry and about the new software economy. Sometimes, it is even regarded as the only viable economic model in this field. This new situation, which succeeded the profusion of all the economic models linked to free software and which were explored during the "new economy" period, should be questioned

all the same. As we have seen earlier, companies like *O'Reilly*, but also certain editors of network games, offer not only the software and its sources but charge the subscription service which makes it possible for the user to take part in the game itself. This not only enables them to attract players but also to profit from their contributions to the development of their program. They also continue to explore new paths, some quite successfully[4].

But this observation does not stop here and, in fact, in the case of free software, these bundle models are mainly "layer" models[5]. The principal impact of free software is to be found at the "lower level" of these layers which deal with the operating systems and, generally, with what is commonly known as "middleware". This can range from application servers to development tools, as well as languages (cf. Figure 1). There is a fundamental reason for this which has to do with the production of free software itself. Free software programs are produced by communities of developers to respond to their own needs. These needs are to be found at the level of the tools and configuration of the machines, in particular that of servers. These are precisely the levels we have just mentioned. It is quite clear that free software programs exist in other fields, but, in the case of middleware, there is a particularly strong link between the motivations of the developers and the nature of the tools which are produced. They are useful to and meant for the developers. As a consequence, when the middleware is free, it can be associated in a bundle with other services. Whenever this is legally possible, it can also be associated either with proprietary software solutions ("upwards" bundle), or with material or hardware ("downwards" bundle)[6].

Figure 12.1 — The various "layers" of computer industry.



In this last case, the concerned participants are the manufacturers. The main manufacturers, IBM but also HP, Sun or Bull in France, have become "sponsors" or "promoters" of free software. This is not a new economic model per se, but rather an extension of their own economic models based on the type of relationship they have established with large software publishers, that is to say, whether or not they depend on them. The role of these promoters, who are becoming the real "bosses" in the historical sense of the word, that is to say, the financiers of both the communities and the free software industry, would undoubtedly deserve to be more clearly explained. Unfortunately, we were not able to do so within the framework of this project. Once again, if this has to do with bundles, the power struggle in which they are locked regarding the free software industry and the communities which develop them, is indeed potentially ambiguous considering

[4]OSDN, VA proprietary software on-line service (ex-VA Linux), which includes slashdot in particular, is another example of this type.

[5]Many thanks to Hughes Rougier and François Horn for helping me to better understand these questions.

[6]As for the middleware, one must also take into account some of the bundles which exist between free software and proprietary software solutions, primarily when the latter are in fact development tools. For example, collaborative development platforms have been edited as proprietary platforms by Collab.Net and by VA Linux, now that it has become VA Software. The Linux development itself uses another similar proprietary software, BitKeeper. Some similar "upwards" bundles have emerged between free solutions and proprietary solutions (we mentioned the example of Suse and Microsoft Offi ce a little earlier), but these attempts remain the exception rather than the rule.

their size and position within their markets. One of the principal questions deals with the extent of the investments that are undertaken and the fact that they can be dedicated to a particular material platform: in other words, this means that a software program can more or less become a proprietary type of software, even if it is free or open. It also means that a de facto standard can be imposed with regard to the hardware.

From this point of view, a rather general manner of interpreting the relationships between companies which devote themselves to developing free software programs and communities of developers (this does not only apply to builders) consists in considering that these collaborative relationships, where a certain amount of opportunism exists (defection of the company but also of the community which, for example, would choose to follow a harmful course for the company) are generally regulated by mutual "hostage" exchanges (Williamson, 1996). As a rule, these hostages are, on the one hand, the developers who were hired by these companies, and, on the other hand, the reputation of these companies which can be easily tarnished by the community if they adopt strategies which move them away from the open-source domain[7]. However, such reasoning does not work for multinationals whose size and market shares are so enormous that they are practically invulnerable. In this case, there is no question of exchanging "hostages" anymore. For these firms, developing products and services dedicated to free software programs means seizing a market opportunity and it is quite obvious that these opportunities will be exploited for profit. This, in turn, will often imply, when these firms are also computer manufacturers, higher specifications in order to meet its material platforms. This, of course, is done so as to increase efficiency but also to intensify a comparative advantage. Risks of opportunism on behalf of these companies thus exist and, contrary to what can occur in smaller size companies, are not really regulated by the existence of hostages.

12.3 The increased role of the (new) integrators.

In any event, the principal role SS2Ls play as promoters is that of integrators. This means that they propose customized solutions to their customers[8]. These solutions can include a complex bundle of services (development, maintenance, information management, etc), of software elements (specific developments, software packages, open-source solutions, etc), of middleware and hardware. These complex goods and services bundles correspond precisely to the customers' complex needs, in particular those of professionals. In this context, the emergence of free software programs corresponds to the emergence of "new integrators" that can integrate without having to maintain special relationships with a particular editor. It also corresponds to an additional integration development model. For the time being, large SSIIs which are not tied to manufacturers have not made a stand with regard to this upcoming battle about integrators, probably because of the close bonds that exist between them and influential editors. Yet, it is not impossible that their interests might bring them closer to open-source software programs. Somehow, they may even be less dependent on these programs than the manufacturers themselves.

It is also within the context of these upcoming battles for customer control that one might interpret the changes being experienced by many companies relating to free software programs, as a move towards what one might call an "specialized integrator" model, however paradoxical this expression might seem at first. Red Hat may be an example: it is as much a distributor as it is a SS2L, but it specifically focuses on various free software programs. Some SS2Ls specialize on software programs like Sendmail or Zope, for example. Some builder-integrators, like IBM, associate their trademark with Linux. In all cases, these are always economic models based on bundles, but which are more sophisticated, where one's specialization and public image play a highly significant role in the competition which is, in most cases, that of a niche market.

[7] Once again, OSDN is the principal hostage or rather "guarantee" that VA Software can still offer to the communities of developers, since SourceForge has now almost become a proprietary software.

[8] See IBM's announcements on the development of "on-demand" services (end of 2002).

IBM and Red Hat might be exceptions since they are "Linux"-specialized integrators. These specialized integrators offer their customers services that evolve "around" one or more software programs, but, in reality, they "cast their net wide". To put it simply, the level of support that is offered for other software programs does not come anywhere near this. This is where the quality of the services, as well as adapted signaling, makes the difference (Jullien, 2001). Otherwise, customers would be attracted by "non-specialized" integrators which would offer a complete, integrated solution to their complex problems. As a consequence, this is also an attempt to segment the market.

At any rate, the period we are entering is that of integrators, whether they are specialized or not[9]. In a way, IT's future resembles what the car industry is going through today. This is a world where "integrators" and "modularity" are the keywords. This can be explained by the fact that integrators try to assemble modules in order to provide customized solutions for their clients. In more general terms, this evolution is seen as a real break with the traditional "large company"(cf. Pavitt [2002]), within an economic context where manufacturing tends to be moving towards countries where labor is cheap. In developed economies, however, the added value is closely tied to the design and development of increasingly sophisticated products. Thus, large car manufacturers have become integrators, who maintain some specific key in-house assets, and who deal with subcontractors for all the other modules. As we have just seen, this is the way things are heading as far as IT integrators are concerned. This is particularly true for the "new integrators" which profit from the emergence of free software programs: these IT designers offer customized services to their clients, with all the "options" they wish to have, and this goes much further than the simple color of their vehicle. These services associate and integrate modules: some will be hardware and others will be software programs (some of which will be open-source programs).

12.4 The second IT modular revolution.

All of this is of paramount importance since, in the world of software programs, a "modular revolution" has accompanied these developments (Langlois [2001]). In a way, as far as IT is concerned, this is the second revolution that has occurred. The first one took place in the field of hardware components. It highlights the supply of services, the design of new and varied products thanks to the assembling of components. First and foremost, this modular revolution was made possible thanks to the existence of software components with open sources. Indeed, this opening alone allows for the easy adaptation of these components, whatever the needs or situation might be, whether it has to do with a specific hardware platform or with a customer's specific needs. Closed software programs, whatever the degree of coding and specification of the interfaces, do not really allow for this type of software and/or hardware solution development and "manufacturing". Market dynamics even tend to make interfaces proprietary and vehemently oppose this sort of evolution[10]. Open-source components are not only easily interoperable, they are reliable. They can also be durably maintained since their sources are very largely accessible. A community of developers as well as specialized companies also add their support.

These companies can obviously be specialized integrators, which would then act as subcontractors for more general integrators. They would thus represent a real interface between the latter and the communities and would be regarded as dedicated and quasi-in-house work markets. But it is also within such a context that one must interpret the emergence of "quasi-editors" for open-source modules (for example, MySQL). They use "double license" models so that they can distribute their products either as free software, for

[9] Of course, this does not stop the coexistence and competition that exists between free and proprietary software solutions (Dalle and Jullien [2003]).

[10] In order to remove this obstacle, we can also interpret the appearance of models, whereby proprietary editors and integrators "share" sources. These models follow a modular economy where the solutions are also components.

noncommercial applications, and/or as a module which brings royalties, whenever it is used within the framework of a proprietary development or integration. In a word, they are then in a position to sell a component - no longer a bundle but an element of a bundle - to companies whose economic model precisely consists in selling bundles. This is a truly "off the shelf " component insofar as its design is not defined by a principal: it corresponds to one or more functionalities which can be integrated in a various number of applications and solutions thanks, once again, to the adaptability of the software goods when they are available in the form of source codes.

In a world of integrators, these quasi-editors can ensure a substantial income and profit from margins which may be more significant than those of specialized integrators. Indeed, they are bound to become subcontractors for more general integrators and to do customized work. Yet, many other criteria (i.e. the amount of risk involved) will undoubtedly be considered in choosing a specific integrator model (specialized or quasi-editors), such as cash-flow, but also the companies' specific backgrounds. These two economic models are mutually supportive since it is highly probable that quasi-editors will offer specialized integrator services[11].

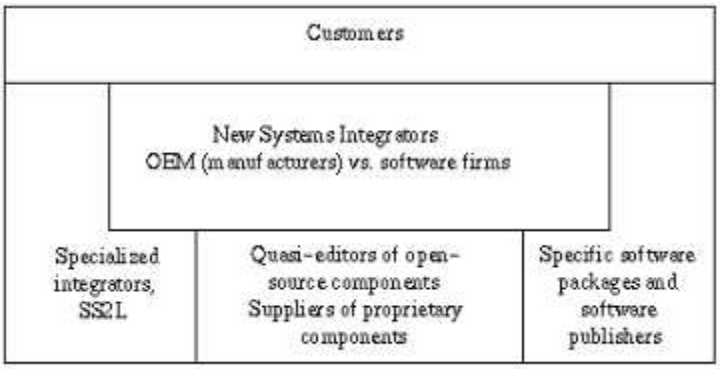
Will IT system integrators then be "genuine" integrators, oriented towards services and integrating hardware or OEMs, that is to say, builders who would use software components which can be adapted to their material platforms? One criterion is undoubtedly linked to the variety of customer requirements. If they can organize themselves around a relatively limited number of architectures where the hardware dimension is significant, then OEMs will probably have a preeminent role. As a consequence, they could, like car manufacturers, have a more significant role in the definition and design of components (hardware, of course, but also software), that quasi-editors could provide them with. To a certain degree, they would also call for integrators that are specialized subcontracting. Inversely, and assuming that architectures would be truly variable and adapted to customers' requirements, the market could favor integrators that focus on services. This means that they use integrators that are specialized in subcontracting and would definitely use "off the shelf "components since they are less specific. Software goods being as adaptable as they are, this second model is less expensive than it may seem at first. It thus increases one's chances to impose oneself on the market, although one can probably imagine both a coexistence and competition between these two forms of integrators.

12.5 Conclusion: heading towards a new software economy?

In all events, this might simply be the future of the new software economy: the opening of sources makes it possible to access modular models and thus developing integration. As a consequence, this new software economy is reinforced by the emergence of new economic actors as well as the evolution of existing participants towards economic models involving relatively specialized integrators. Open-source software component quasi-editors have also emerged. This evolution is thus forming the foundation of a new industrial IT organization: a new software economy that focuses on its customers and which is increasingly adapted to their particular needs. This is made possible thanks to the intervention of integrators and specialized integrators which are provided by software package and software component editors, but also thanks to open software component quasi-editors (cf. Figure 2). In a way, the fact that the IT organization was not properly adapted hindered the installation of a certain type of component: the arrival on the scene of open software programs seems likely to provoke major changes in the future that will give rise to developments which will require the progressive installation of a real software component model. This will also entail a renewed industrial organization.

[11] This question undoubtedly arises in a particularly acute way for the firms which are oriented towards strategies of "freeing" software (Horn [2002]).

Figure 12.2 — The new software Economy.



Within this new software economy, it would be an illusion to think that "manufacturers" will be absent (von Hippel [2001]). This illusion is due to the fact that software goods have particular characteristics. There are no production lines per se in this field. Moreover, they are part of a bundle economy where hardware, and more specifically, services, play an essential role. On the contrary, the emergence of the free software industry, the development of components and integrators, would tend to reinforce manufacturing processes in the IT as well as the software sector. In a way, they are much more integrated and much more organized. If well-informed users can do without suppliers and benefit from the work of other users, a great number of professional customers, if not to say the majority, will choose the disintegration of the less "specific" aspects of IT (Williamson [1985]).

To summarize, one should not confuse industrial organization with innovation. These two questions complement each other, they are not limited to one another. In the world of software, if the question of innovation is indeed that of its "sources" (von Hippel [1988]; Dalle [2002]), it is within this framework that users clearly have a significant role to play. Yet, system integrators (Pavitt, 2002), whether they are specialized or not, are the ones that meet the users' needs. They are the ones that will need to ensure the *manufacture* of an IT solution that will be adapted to each request and based on software components, whether open-source and/or proprietary. We are obliged to accept that the emergence of the free software industry has not allowed this convenience to be distributed like a raw material, contrary to what had been expected at one point. The very nature of software goods and in particular their adaptability, somehow goes against this simplistic vision of things. It points towards a richer and completely different industrial organization, where the work of transformation and adaptation is generally done by integrators rather than by the users themselves.

Bibliography

G. AHARONIAN, 2000. Patent examination system is intellectually corrupt. URL: http://www.bustpatents.com/corrupt.htm.

. AMBERLAB, Mose. ..... URL: http://amberlab.net/ubp/.

W. B. ARTHUR, 1989a. Competing technologies, increasing returns and lock-in by historical events: The dynamics of allocations under increasing returns to scale. *Economic Journal*, 99:116–131.

C. G. AU PLAN, 2002. Économie du logiciel: renforcer la dynamique française. rapport du groupe présidé par Hugues Rougier, Rapporteur Général: Jean-Michel Dalle, Rapporteur: Sylvie Bénard.

A. BERTRAND-DOULAT, 1997. Les dessins et modèles. In M. VIVANT, editor, *Les créations immatérielles et le droit*. Ellipses.

J. BESSEN, 2002. Open source software: Free provision of complex public goods. Research on Innovation. URL: http://www.researchoninnovation.org/online.htm#oss.

C. BESSY and E. BROUSSEAU, december 1998. Technology licencing contracts: features and diversity. *International Review of Law and Economics*, 18:451–489.

C. BESSY and E. BROUSSEAU, 2001. Contrats de licence et innovation. In P. MUSTAR and H. PENAN, editors, *Encyclopédie de l'innovation*. Economica, Paris.

A. BEUGNARD, 2001. Nouvelle économie du logiciel... un point de vue. projet «Nouvelle économie du Logiciel», RNTL. URL: http://www-eco.enst-bretagne.fr/Etudes_projets/RNTL/workshop1/Beugnard.pdf, 1^{re} réunion de travail.

A. BOUSQUET and M. WOLKOWICZ, 1^{e} trim. 1994. Les enjeux économiques dans les contrats de licence. *L'écho des recherches*, 155.

R. BOYER and M. FREYSSENET, 1995. Émergence de nouveaux modèles industriels. hypothèses et démarches d'analyse. Technical Report 15, actes du GREPISA.

R. BOYER and M. FREYSSENET, 2000. *Les modèles productifs*. Collection Repères, La Découverte, Paris.

P. BRETON, 1990. *Une histoire de l'informatique*. Point Sciences, Le Seuil, Paris.

. CARDIONEWS, Mose. Site de l'association de cardiologues cardionews. URL: http://www.cardionews.com/.

M. CASTELLS, 1998. *La Société en Réseaux*. Fayard.

M. CLÉMENT-FONTAINE, 1999. Étude de la gnu gpl. Msc Dissertation, Université de Droit, Montpellier. http://crao.net/gpl/gpl.html.

W. M. COHEN and D. A. LEVINTHAL, 1989. Innovation and learning: The two faces of r&d. *Economic Journal*, 99:569–596.

P. COHENDET, F. CREPLET, and O. DUPOUET, 2002. Innovation organisationnelle, communautés de pratique et communautés épistémiques: le cas de linux. *Revue française de gestion*, to be published.

P. COHENDET, F. CRÉPLET, and O. DUPOUET, 2001. Interactions between epistemic communities and communities of practice as a mechanism of creation and diffusion of knowledge. In J.-B. ZIMMERMANN and A. KIRMAN, editors, *Interaction and Market Structure*. Springer, Londres.

C. COLOMBET, 1999. *Propriété littéraire et artistique et droits voisins*. Précis Dalloz.

M. CORIS, 2002. Les sociétés de services en logiciels libres. l'émergence d'un système de production alternatif au sein de l'industrie du logiciel? In JULLIEN ET AL. [2002], editor, *Nouveaux modèles économiques, nouvelle économie du logiciel*, pages 89–106.

G. CORNU, 1987. *Vocabulaire juridique*. Association Henri Capitant, P.U.F.

J.-M. DALLE, 2002. Open-code: The sources of open-source innovations. In L. DAVIS, editor, *The Changing Role of Intellectual Property Rights*. Economics of Innovation and New Technologies. to be published.

J.-M. DALLE and N. JULLIEN, 2000. Nt vs. linux, or some explorations into the economics of free software. In G. BALLOT and G. WEISBUCH, editors, *Applications of Simulation to Social Sciences*, pages 399–416. Hermès, Paris.

J.-M. DALLE and N. JULLIEN, january 2003. 'libre' software: Turning fads into institutions? *Research Policy*, 32(1):1–11.

P. A. DAVID, may 1985. Clio and the economics of qwerty. *American Economic Review*, 75-2 Papers and Proceeding:332–337.

J. DE BANDT, 1995. *Services aux entreprises: informations, produits, richesses*. Economica, Paris.

P. B. DE LAAT, 2000. Patenting mathematical algorithms: What's the harm. *International Review of Law and Economics*, 20.

M. DELAPIERRE, L.-A. GERARD-VARET, and J.-B. ZIMMERMANN, december 1980. Choix publics et normalisation des réseaux informatiques. Technical report, Rapport BNI.

M. DELAPIERRE and J.-B. ZIMMERMANN, 1994. La nature du produit informatique. *Terminal*, 65, automne: 87–104.

G. DRÉAN, 1996. *L'industrie informatique, structure, économie, perspectives*. Masson, Paris.

Y. DUPUIS and O. TARDIEU, 2001. Les brevets superflus en matière de logiciel. report, École des Mines, Paris.

. EUROSTAF, 2000. L'informatique: le passage d'une logique de produit à une logique de services. Technical report, Eurostaf.

J. FARRELL and G. SALONER, 1988. *The Economics of Converters*. MIT.

B. FAUCON and J.-P. SMETS-SOLANES, 1999. *Logiciels libres. Liberté, égalité, business*. Éditions Edispher, Paris.

D. FORAY, 2^{e} trimestre 1989. Les modèles de compétition technologiques, une revue de la littérature. *Revue d'économie industrielle*, 48:16–34.

D. FORAY, 1[er]trimestre 1990. Exploitation des externalités de réseau versus évolution des normes: les formes d'organisation face au dilemme de l'efficacité, dans le domaine des technologies de réseau. *Revue d'économie industrielle*, 51:113–140.

D. FORAY, 2002. Innovation et concurrence dans les industries de réseau. *Revue Française de Gestion*, pages 131–154.

D. FORAY and J.-B. ZIMMERMANN, october 2001. L'économie du logiciel libre: organisation coopérative et incitation à l'innovation. *Revue économique*, 52:77–93. Special Issue, hors série: économie d'Internet, sous la direction d'É. Brousseau and N. Curien.

J. FOYER and M. VIVANT, 1991. *Le droit des brevets*. Thémis, PUF, Paris.

J. GADRAY, 1998. La caractérisation des biens et des services, d'adam smith à peter hill: une approche alternative. Technical report, IFRESI, Lille. document de travail.

N. T. GALLINI and R. A. WINTER, 1985. Licensing in the theory of innovation. *RAND Journal of Economics*, 16(2).

C. GENTHON, 1995. *Croissance et crise de l'industrie informatique mondiale*. Syros, Paris.

C. GENTHON, 2000. Le cas sun microsystem. ENST Bretagne. URL: http://www-eco.enst-bretagne.fr/Enseignement/2A/1999-2000/EST201/sun/sun00.htm, support de cours.

C. GENTHON, 2001. Le libre et l'industrie des services et logiciels informatique. projet «Nouvelle économie du Logiciel», RNTL. URL: http://www-eco.enst-bretagne.fr/Etudes_projets/RNTL/workshop1/genthon.pdf, 1[re]réunion de travail.

J.-Y. GOFFI, 1996. *La philosophie de la technique*. Que sais-je?, PUF, Paris.

L.-A. GÉRARD-VARET and J.-B. ZIMMERMANN, may 1985. Concept de produit informatique et comportement des agents de l'industrie. In *colloque «Structures économiques et économétrie»*, may 1985.

. GUIDE LAMY, 2000. *Droit de l'Informatique*, volume 3415. Guide Lamy.

P. HIMANEN, 2001. A brief history of computer hackerism. URL: http://www.hackerethic.org/writtings/hackerhistory.shtml.

F. HORN, 1999. L'importance du logiciel libre dans l'amélioration de l'efficience des logiciels. *Terminal*, 80/81:119–148. Special Issue, *Le logiciel libre*.

F. HORN, 2000b. *L'économie du logiciel. Tome 1: De l'économie de l'informatique à l'économie du logiciel. Tome 2: De l'économie du logiciel à la socio-économie des «mondes de production» des logiciels*. PhD, Université de Lille I, mention: économie industrielle, 570 pages. URL: http://www-eco.enst-bretagne.fr/Etudes_projets/RNTL/documents_universitaires.html.

F. HORN, 2002. Les stratégies de libération du code source d'un logiciel par une entreprise: opportunités et difficultés. In JULLIEN ET AL. [2002], editor, *Nouveaux modèles économiques, nouvelle économie du logiciel*, pages 107–128.

N. JULLIEN, 1999. Linux: la convergence du monde Unix et du monde PC. *Terminal*, 80/81:43–70. Special Issue, *Le logiciel libre*.

N. JULLIEN, november 2001. *Impact du logiciel libre sur l'industrie informatique*. PhD, Université de Bretagne Occidentale / ENST Bretagne, mention: sciences économiques, 307 pages. URL: http://www-eco.enst-bretagne.fr/Etudes_projets/RNTL/documents_universitaires.html.

N. JULLIEN, M. CLEMENT-FONTAINE, and J.-M. DALLE, 2002. «nouveaux modèles économiques, nouvelle économie du logiciel». rapport final. Technical report, Projet RNTL, 218 pages. URL: http://www-eco.enst-bretagne.fr/Etudes_projets/RNTL/.

M. L. KATZ and C. SHAPIRO, august 1986. How to license intangible property. *The Quaterly Journal of Economics*, 406:567–589.

M. KOTABE, A. SAHAY, and P. S. AULAKH, 1996. Emerging role of technology licensing in the development of global product strategy: conceptual framework and research propositions. *Journal of Marketing*, 60:73–88.

K. LAKHANI and E. VON HIPPEL, 2000. How open source software works: Free user to user assistance. URL: http://opensource.mit.edu/online-pap-abst.html, travail de recherche.

R. N. LANGLOIS, 2001. The vanishing hand: the modular revolution in american business. mimeo.

J. LERNER and J. TIROLE, 2002. Some simple economics of open source. *Journal of Industrial Economics*, 52. URL: http://opensource.mit.edu/online-pap-abst.html. to be published.

. LES ÉTUDES DU CONSEIL D'ÉTAT, 1998. *Internet et les réseaux numériques*. La Documentation Française, Paris.

X. LINANT DE BELLEFONDS, january 2001. Brevetabilité du logiciel? à revoir. *Communication - commerce électronique*.

january 2002a. *IBM-Linux: une réalité économique*, Paris. Linux Expo 2002. conference.

january 2002b. *The Inevitable Success of the Open Source Technology Model*. Linux Expo 2002, Paris. Conférence.

january 2002c. *Un business model gagnant pour un éditeur de logiciel libre*. Linux Expo 2002, Paris. conférence.

A. LUCAS and H.-J. LUCAS, 2001. *Traité de la propriété littéraire et artistique.*, volume 419. Litec (2^d ed.).

. MANDRAKESOFT/BOURSORAMA, 2002. Suivre le cours en bourse de MandrakeSoft. URL: http://www.boursorama.com/cours33.phtml?symbole=1r04477-OTC.

J. MATEOS GARCIA, 2001. Innovating without money. Linux and the Open Source paradigm as an alternative to commercial development. Msc Dissertation, SPRU, University of Sussex.

. MATHÉLY, 1991. *Le nouveau droit français des brevets d'invention*. Librairie du journal des Notaires et des Avocats, Paris.

D. MAUME, 7 janvier 2002. Linux s'impose dans les entreprises. *L'Usine Nouvelle*.

MOSE, 2002. Site de Makina Corpus dédié à la Communauté. URL: http://www.makina-corpus.org/.

A. MOULINE, 1996. *L'industrie des Services Informatiques*. Economica, Collection Économie Poche, Paris.

K. PAVITT, june 2002. System integrators as «post-industrial» firms? In *Industrial Dynamics of the New and Old Economy - who embraces whom?* DRUID Summer Conference, june 2002.

F. POLLAUD-DULLIAN, 1999. *Droit de la propriété industrielle*, volume 238. Montchrestien, Paris.

A. R., 4 Juillet 2001. Pour les ingénieurs d'alcôve, le temps aussi est libre. *Les Echos*.

E. S. RAYMOND, 1998a. The Cathedral and the Bazaar. URL: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/.

E. S. RAYMOND, 1998b. Homesteading the Noosphere. URL: http://www.tuxedo.org/~esr/writings/homesteading/.

E. S. RAYMOND, 1999. *The Cathedral & the Bazaar; Musing on Linux and Open Source by Accidental Revolutionary*. O'Reilly, Sebastopol, Calif.

D. K. ROSENBERG, 2000. *Open Source. The unauthorized white papers*. IDG Books Worldwide, inc.

. ROUBIER, 1952. *Le droit de la propriété industrielle, Tome 1*. Recueil Sirey, Toulouse.

. ROUBIER, 1954. *Droit de la propriété industrielle*. Recueil Sirey, Paris.

C. SHAPIRO and H. VARIAN, 1999. *Information Rules: A Strategic Guide to the Network Economy*. Harvard Business School Press.

C. SHAPIRO and H. VARIAN, 2000. *Économie de l'information - guide stratégique de l'économie de réseau*. De Boeck Université. Traduction française de Shapiro and Varian [1999].

P. SIRINELLI, 1992. *Propriété littéraire et artistique et droits voisins*. Mémentos Dalloz.

J.-M. THIÉRY, 1999. Quatre décennies de logiciels et de données scientifiques libres. *Terminal*, 80/81: 191–202. Special Issue, *Le logiciel libre*.

M. VIVANT, editor, 1997. *Les créations immatérielles et le droit*. Ellipses, Paris.

M. VIVANT, July 2002. Intervention. In *Propriété intellectuelle et mondialisation*. ERCIM, to be published, July 2002.

M. VIVANT and . AL., 2001. *Droit de l'Informatique*, volume 192. Guides Lamy.

M. VIVANT and C. LE STANC, 2002. *Droit de l'Informatique*, volume 151. Guides Lamy.

F. VON GOTTL-OTTLILIENFELD, 1914. *Précis d'économie sociale*, chapter Économie et technique. .

E. VON HIPPEL, July 1986. Lead users: a source of novel product concepts. *Management Science*, 32(7).

E. VON HIPPEL, 1988. *The Sources of Innovation*. Oxford University Press, New York.

E. VON HIPPEL, 2001. Open-source shows the way: Innovation by and for users - no manufacturer required! *Sloan Management Review*, été.

M. WEBER, 1971. *Économie et société*. Plon, Paris.

D. A. WHEELER, 2001. Why Open Source Software/Free Software (OSS/FS)? Look at the Numbers. URL: http://www.dwheeler.com/.

O. E. WILLIAMSON, 1985. *The Economic Institutions of Capitalism*. Free Press, New-York.

J.-B. ZIMMERMANN, 1ertrimestre 1989. Groupes industriels et grappes technologiques. *Revue d'économie industrielle*, 47(5).

J.-B. ZIMMERMANN, september 1995a. Le concept de grappes technologiques. Un cadre formel. *Revue économique*, 46(5):1263–1295.

J.-B. ZIMMERMANN, 1995b. L'industrie du logiciel: de la protection à la normalisation. In M. BASLE, D. DUFOURT, J.-A. HÉRAUD, and J. PERRIN, editors, *Changement institutionnel et changement technologique. Évaluation, droits de propriété intellectuelle, système national d'innovation*, pages 195–207. CNRS Éditions, Paris.

J.-B. ZIMMERMANN, 1998. Un régime de droit d'auteur: la propriété intellectuelle du logiciel. *Réseaux*, 88-89:91–106.

J.-B. ZIMMERMANN, 1999. Logiciel et propriété intellectuelle: du copyright au copyleft. *Terminal*, 80/81: 95–116. Special Issue, *Le logiciel libre*.