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

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

LEM
Working Paper Series

Why open source software can succeed

Andrea Bonaccorsi[*]
Cristina Rossi [†]

[*] *Sant'Anna School of Advanced Studies*
[†] *Sant'Anna School of Advanced Studies*

2002/15 **July 2002**

ISSN (online) 2284-0400

Why open source software can succeed

Andrea Bonaccorsi[*], Cristina Rossi

Laboratory of Economics and Management
Sant'Anna School of Advanced Studies
Pisa, Italy

Abstract

The paper discusses three key economic problems raised by the emergence and diffusion of Open source software: motivation, coordination, and diffusion under a dominant standard.
First, the movement took off through the activity of a software development community that deliberately did not follow profit motivations. Second, a hierarchical coordination emerged without the support of an organization with proprietary rights. Third, Linux and other open source systems diffused in an environment dominated by established proprietary standards, which benefited from significant increasing returns.
The paper shows that recent developments in the theory of *critical mass* in the diffusion of technologies with network externality may help to explain these phenomena.

Keywords: Open Source, Diffusion, Network Externality

[*] Address: P.zza Martiri della Libertà 33, 56127, Pisa; Tel.: +39 050 883323; Fax: +39 050 883344. *E-mail address:* bonaccorsi@sssup.it

Introduction

As one of the founding fathers of the Open Source phenomenon has observed "Linus Torvalds and his companions [have supplied] hundreds of megabytes of programmes, documents and other resources, even entire suites of Internet tools, desktop publishing, graphics and editing programmes, games and so on" (Raymond, 1999, 224) and "in the Web Server market the 20% of Microsoft pales into insignificance when compared to the 53% of Apache" (Di Bona, Ockman, Stone, 1999, 9). Yet such a massive phenomenon seems to challenge a number of entrenched economic ideas.

From an economic point of view the introduction of Open Source software (OSS) can be analysed as an important process innovation in the software production process as opposed to the traditional property-based approach of the commercial world. This phenomenon raises a series of important questions that, for reasons of clarity, can be formulated in terms of Schumpeterian three-phase axis of the innovative process (Schumpeter, 1934). As it is well known, the innovative process involves the birth of a new idea (invention), its economic application through transformation into a new product or process (innovation) and its subsequent introduction among consumers and businesses (diffusion). For each of these phases the Open Source phenomenon generates extremely interesting theoretical puzzles for economic analysis, which can be formulated as follows:

(a)Why do programmers write Open Source codes if no one pays them to do it?

(b) How do hundreds of people spread around the world manage to effectively coordinate with each other in producing programmes consisting of millions of lines of code[1] in the absence of any hierarchical structure based on the ownership of assets?

[1] According to J. Sanders (1998), in 1998 the kernel of the Linus operating system comprised over one and a half million lines of code. In the first version, implemented by Linux Torvalds in 1991, there were 10,000 lines.

(c) Why is it that Open Source programs are becoming so widespread in a world dominated by Microsoft-imposed standards?

The first question regards the invention phase: there is no problem in explaining an individual inventing the concept of free software and proposing it to the programming community, but that still leaves unexplained the phenomenon of hundreds of individuals continually inventing programs which are then made available free of charge. What system of incentives regulates the initial innovation phase? What motivates such behaviour?

The second question concerns the problem of coordination, which is crucial for the transformation of an invention into economically advantageous and marketable innovations. The spontaneous, decentralized functioning of the Open Source community represents a challenge to many current notions of coordination. How is it possible to align the incentives of several different individuals without resorting to property rights and related contracts? How is it possible to specify everyone's tasks without resorting to a *formal* hierarchical organization?

Finally, the third question focuses on the conditions for the diffusion of Open Source software in the population of users. How is it possible to increase the diffusion of new technology given the presence of a well-established standard, and thus of increasing returns from the adoption of the previous technology?

In case of network technologies, a large part of the diffusion theory forecasts the emergence of a dominant standard and, as a second instance, the coexistence of different standards only if the diffusion dynamic of the competing technologies starts at the same moment. On the contrary, the Open Source software, spreads across the system although there is a dominant standard.

The aim of this paper is to explain what at first sight would seem "non sense" by combining recent developments in economic analysis such as the theory of collective action and the theory of technological diffusion in the presence of network externalities. It will become clear how a non-conventional understanding of the phenomena can provide a plausible explanation.

At the same time it will become clear how the free software is fated to cohabit with the proprietary one, but also to originate an articulate series of new organizational forms (contracts, licences, firms) in which open and proprietary elements will combine variously.

1. Open Source software and individual motivations: the point of view of the economic theory

Stallman proposed a revolutionary idea in 1984 with the "Free Software Foundation", subsequently confirmed in 1997 in the "Open Source Definition[2]". The key concept is that there should be unrestricted access to computer programming codes: anyone should be able to use them, modify them and circulate such modifications without having to pay anything.

According to the theorists of the new development model, called *bazaar* (Raymond, 1999), when programmers are allowed to work freely on the source code of a program, exchanging files and ideas through the Internet network, this will inevitably be improved because collaboration helps to correct errors and enables adaptation to different needs and hardware platforms[3]. In this way quite complex programs can be written down efficiently without a central authority in control.

Developers from all around the world work asynchronically on the various Open Source projects, often during their spare time and week ends. The hierarchically organized and top down planned structure adopted in most productive processes is abandoned in favour of a new kind of bottom up structure, which is non-coercive and largely decentralized, even if shared behavioural rules are still present.

This makes the Open Source software a fearful competitor for the incumbent firms of the software market. These firms, in fact, face a threat coming from the outside: it is not a new competitor that makes things in the same way but more quickly and efficiently, but rather there is a new paradigm contrasting the traditional one. According to the classic model of

[2] Open Source Definition, http://www.opensource.org
[3] Introduction to Open Source, http://www.opensource.org

development, a handful of programmers paid for their work have the task of writing a code, which is then protected by copyright. This licence will allow the distribution of the code only in the binary form and with strong use limitations that will forbid any duplication and modification.

Moreover, Open Source developers are not subject to the pressures that in software house are induced by corporate announcements of more up-to-date and efficient releases.

All these elements seem to lead to software of higher-quality.

Reliability, in fact, is numbered amongst the characteristics that make Linux "an easy sell". In comparison to Windows NT, it has a resistance to crashing that can be measured in "months and years rather than days or weeks". The Apache web server equipped with the Linux Red Hat 4.2 operating system and Apache, installed in 1997 in the laboratories of the Compaq Computer Corporation, resisted 267 times before crashing and even then it only did so following a massive storm that led to a total power cut in the building where it was housed. The operating system designed by Torvalds is also highly portable: "Nowadays Linux runs on anything, from the Palm Pilot to the Alpha workstation, and is the operating system most subject to porting amongst those available for PC" (Torvalds, 1999, 112).

The first urgent question is therefore the one clearly put by Glass (1999, 104): "I don't know who these crazy people are who want to write, read and even revise all that code without being paid anything for it at all". The motivations of Open Source developers, deeply analysed by the founder fathers of the movement, have started to interest also the economic theory as the phenomenon went out from universities and research centres, became "tremendously successful" (Lerner e Tirole, 2001, 819) and create new and often lucky business models.

Raymond, one of the most famous Open Source *evangelist*, explicitly establish a connection between the incentives of open source programmers and *hacker culture* values. Open Source developers are considered the heirs of the "Real Programmers" of the immediate post-war period, who "came from the engineering and physics fields…[and] programmed in FORTRAN and another half-dozen languages that have now been forgotten" (Raymond, 1999).

Hackers, who are basically computer scientists, come prevalently from the academic world or from the research centres of large software corporations – MIT's Artificial Intelligence Lab in Boston, the Laboratory of Artificial Intelligence at the University of Stanford, the Computer Science Department at Berkeley, the Xerox Park, to name just a few.

Hackers regard programming as an *art form* and feel "an artistic satisfaction" in writing code. Such a satisfaction is quite similar to the one that music gives to a composer and painting gives to an artist.

However some well-known myths about free software need to be reappraised. A study on one of the luckiest Open Source project shows that not all programmers work for free in a totally anarchic framework having the file exchange and the user group on the Internet as only coordination forms.

A survey of the Institut fuer Psychologie at Kiel University, in Germany, (Hermann, Hertel, Niedner, 2001), made on a sample of 141 developers of the Linux kernel, underlines that more than 43% of the programmers receives some form of income while 20% receives a salary "for their Linux programming on a regular basis". At the same time, in the largest projects there is always a "well-respected leader" (Lerner e Tirole, 2001, 823) or a "core development group" (Mockus et al.2000) having the task of singling out the main problems to solve, choosing among the various solution proposed by the programmers and deciding the official release of the code produced by the working group. Several authors (Bezrukov, 1999) observe as the organisation structure of the Linux kernel developer group is hierarchically organized around the charismatic leader Linux Torvalds who acts as a "benevolent dictator". Moreover not always there are thousand developers: the Linux kernel mailing list has over than 3500 members but the real contributors are less than 500.

Nevertheless all this does not deny the importance to deeply understand the structure of the incentives of people working on Open Source projects. Sociology and social psychology number the hedonistic (or playful) motivation among the determinants of taking part in whatever social movement, while altruism is considered as lying at the basis of a lot of human

behaviours. The so called *"gift economy"* (Mauss, 1959) states that giving away goods and services for free allows to make and maintain social links and implies the duty of reciprocate. This happens also when the exchange is not in favour of well-known individuals but of a community of unknown subjects. In fact, people hope to have help and support in future given that the previous contributions have created a tacit reciprocity agreement.

A careful analysis shows, however, that it is not necessary to assume altruistic behaviour, but rather it is possible to reduce the motivational problem in Open Source to a classical cost-benefit framework, giving the right prominence to the constituent elements. On the cost-side, we observe that the infrastructure investment for taking part in the movement is almost zero. Most of the programmers have their own computer for studying or working or even write the code at work. The above mentioned survey about Linux kernel developers finds 38% individuals who write Open Source code at work also if this is not an official duty. In the same way Internet connection is no more a problem for most of them. The presence of contributes from all over the world is not much more than a romantic idea: most of the LOCs added to different projects are from European or United States developers, that clearly have no connectivity problems. Given that, the main investment involves time and intellectual resources. As regard to this point, several studies underline that investment in time is quite limited too. In an analysis of more than 1700 individuals taking part to an important newsgroup that gives on line help about the Apache Web server, Lakani and von Hippel (2000) find that over than 80% subject answering to the help requests spend an average time of five minutes for giving the solution to the question. The reason of this is quite simple: programmers know the answer as they know the operation of the newsgroups, the main e-mail programs, the programs for file compressing that, reducing file dimension, allow them to circulate easily on the Net. Most of the developers had the possibility to work with Unix from which many Open Source programs come from. All this gives to them the possibility of bearing their competencies easily to the problem at hand.

Non negligible benefits stand against low cost. From the point of view of the economic theory the motivations of intrinsic utility, of gaining a reputation and signalling quality of human capital, and of learning are prominent. Such element have always been the engine of the scientific community.

Open Source software is a form of intellectual gratification with an *intrinsic utility* similar to that of scientific discovery. Emerging as it does from the university and research environment, the movement adopts the motivations of scientific research, transferring them into the production of technologies that have a potential commercial value. The new way of developing code is basically nothing other than "the natural extension of Western scientific culture" (Stone, 1999). The process of scientific discovery involves the sharing of results, just as the dictates of the Open Source movement involve sharing source code. Sharing results enables researchers both to improve their results through feedback from other members of the scientific community and to gain recognition and hence prestige for their work. The same thing happens when source code is shared: other members of the group provide feedback that helps to perfect it, while the fact that the results are clearly visible to everyone confers a degree of prestige which expands in proportion to the size of the community.

A further element is that working on Open Source projects provides the prestige and visibility that often gives programmers the chance to be noticed by software firms. All of the founding fathers of the movement "have earned enough of a reputation to allow them to pay the rent and maintain their kids" (Di Bona, Ockman, Stone, 1999, 14). Not counting that many developers are computer science students that take part in Open Source projects for finding materials for their undergraduate or graduate theses. In this way they develop skills in programming that are going to be useful when they enter the labour market.

The new development paradigm constitutes an immense learning opportunity: having all the code they want, programmers can study it very deeply. Learning from reading the answers to the questions posed by the users is one of the most important motivations for participants to

the newsgroups devoted to the resolution of the problems that arise from Open Source programs (Lakhami, von Hippel, 2001).

In addition to "fun to program"(Torvalds, 2001) and gaining a reputation through competition with other developers in finding the smartest and most efficient software solution, the theorists of the movement single out the benefit of obtaining a software solution which is not already available in the market.

Many Open Source projects take shape because the people promoting them have looked in vain for a program to perform a particular function. They arise, that is, to satisfy a demand for which there is no corresponding supply, in short to "fill an unfilled market" (Green, 1999). One typical example is the Perl language developed by a systems administrator at Burroughs, who saw a need for something combining the speed and immediacy of the Unix shell languages with the possibility to develop more complex programmes, a typical feature of high-level languages like C. Unable to find that "something", he created and then distributed it to the entire Open Source community. Today Perl is a commonly accepted language used above all for Internet-related applications.

Moreover there are several minor motivations. Open Source developers are highly suspicious of the "customerization of the computer" caused by Microsoft (Ullman, 1998). They often think of their contribution as fundamental for project success (instrumentability: Hertel, 2002) and attribute a great importance to project goals.

Even setting the incentive structure in a classical cost-benefit analysis, we have to address the problems of free riding and network externality clearly present in information production.

The Open Source movement have brought about a profound change in the nature of the software good, which has actually assumed the characteristics of a *collective good* (Bessen, 2001). Once produced, these goods can be freely consumed also by those who have not contributed, or only very little, to their production. At first sight, it might seem preferable for everyone to exploit the contributions of others, obtaining benefits without sustaining any cost

whatsoever. However, if everyone followed the same reasoning, the good would not be produced at all.

As it is well known, despite being widely debated in economic theory, total free riding (with all agents choosing to contribute nothing) rarely occurs both in the real world and in laboratory experiments (Fox, Guyer, Hamburger, 1975). In our case, however, what needs to be explained is how self-interested, free riding behaviour is avoided and how sustained cooperation is also maintained in such a *large-scale community*.

A possible explanation starts by considering the software produced outside commercial channels as nothing more than a straightforward instance of the general issue of *collective action* and examining the above-mentioned motivations from this point of view . The size of the phenomenon is extremely interesting from a theoretical point of view, because it seems to contradict the theory, proposed by Olson, of an *inverse* relation between the size of the group and the probability that collective action will be successful. (Olson, 1965). The argument is that as the size of the group increases there is a progressive reduction in the probability of any defection being detected and a corresponding increase in the incentive to benefit from the good without contributing to its production. However, Olson's analysis does not take into account a set of variables whose effect cannot be ignored, above all the heterogeneity of subjects called upon to participate in the collective action. Hardin (1982) underlines the centrality of the heterogeneity of resources and interests within the group: a small set of strongly motivated and resourceful subjects (*small group*) is often capable of supply of collective goods *without* the cooperation of other individuals. In the Open Source movement Hardin's small group, whose presence is essential for the success of collective action, is formed by the hackers. Their major resource is their indisputable know-how in computer science. Their most marked interest lie in their experience of code writing as an intellectual activity, in their work or research needs and also in the desire for gaining reputation and signalling their talent.

Their supply action is also made very effective by the fact that the source code is a non-rival good in terms of consumption. The benefits it brings are invariant with respect to the

number of users: anyone can download code from the Web, compile and use it without reducing other people's capacity to do so. This further confutes Olson's thesis. As Chamberlain (1974) demonstrated in the case of non-rival collective goods, the quantity supplied increases with the size of the group.

Hardin's conclusions provide an account of how Open Source software is produced by the most interested hackers, but it does not explain why other individuals also collaborate in its production in an increasingly extensive process of cooperation. For this it is necessary to refer to Marwell and Oliver's work on the theory of Critical Mass. Initially proposed by Schelling (1978), it was developed in sociology by Marwell and Oliver (1993) and then picked up in economics by Katz and Shapiro (1985), Economides (1995), Witt (1997) and Huberman (1999).

The presence of many highly interested subjects allows to get over the initial phases of supply of the collective good, where cooperation costs overcome benefits. After this phase, more and more agents find out that contribution is profitable for more and more agents. All this sets in motion a virtuous circle that enables the phenomenon to be self-sustaining converging towards the new equilibrium in which all members choose to cooperate.

Here the role of strongly interested subjects is not to provide the good entirely themselves, but to create the necessary conditions for production to become easier. Their task is "to be the first to cooperate", enabling the difficult initial start-up phase of collective action to be overcome. This framework explains very well the dynamics of Open Source projects. They originate from a small group of subjects working for achieving a deliberate given goal. If, in this starting phase, the group succeeds in finding good solutions, the results are posted on the Internet and the project is advertised on mailing lists and newsgroups deciding which problem to work on. At this point, the project has two possible outcomes: attracting and increasing a number of contributors or giving rise to no interest. In the first case a well structured Open Source project is obtained, such as in the Linux case, while in the second one the project is going to die according to the fence of Schelling's (1978) "dying seminar".

2. *Open Source and "non sexy work": in search of hybrid models*

The above mentioned framework explains the development of activities that present some artistic or scientific intrinsic gratification. On the contrary, it is not appropriate for the so-called "non sexy" work involving, for example, the development of graphical interfaces, the compilation of technical manuals, the on line support in newsgroups and so on. According to Lakhani and von Hippel (2001) the motivations for giving on line help are not so different from the ones of programmers solving complex software problems. However, it is difficult to accept the idea that these low gratification activities could be motivated by the same incentive structure than high level, creative work. The incentive structure of low technology value activities is one of the most challenging for a paradigm having creativity and reputation as fundamental values.

Although these activities display a low degree of innovativeness, they are fundamental for the adoption of Open Source software. The two most famous Linux interfaces, KDE and Gnome, created in 1998, provided Linux users with a mouse driven deskpot environment. Such kind of environment, which has become the standard in the commercial world, favoured both the adoption of Linus Torvald's operating system by less skilled subjects and the migration of Windows and Mac users toward it.

Economic analysis of diffusion shows that technological discontinuities are liable for the emergence of new trajectories, but the spread across the system depends on a cumulative series of little incremental innovations displaying, in many cases, no intrinsic technological interest.

The emergence of new hybrid business models seems to solve this kind of problem.

There is, in fact, complementarity between the development of programs by Open Source communities and the emergence of companies and software houses producing free software, giving it away to customers free of charge and shifting the value from licensing agreements to additional services such as packaging, consultancy, maintenance, updating and training. This business model is widely accepted in Open Source communities and is characterised by a

detailed regulation presenting innovative legal solutions which address the relation between free and on payment services.

A number of successful firms adopt variants of this hybrid business model. Among these, Red Hat not only resisted the explosion of the speculative bubble which eliminated from the market a lot of technology-based firms in spring 2000, but continues to make profits. This company, which provided NASA with the software for space missions, thinks of itself as a car producer. In the automobile industry, in fact, different components produced by different firms are assembled for creating a product that satisfies consumer needs. In the same way, Red Hat puts together and compiles the source code of the Linux operating system obtaining a program ready for installation, which is distributed on a CD at less than 100 dollars together with a set of accessory applications (for instance the Gimp program for managing and changing images) and a complete documentation.

Recently both new actors providing support services for Open Source software and incumbents in the commercial software industry imitated Red Hat. In particular the software giants took note of the inevitable coexistence of the two paradigms and decided to make their products compatible with Open Source platforms, to release the code of some programs and even to take part in Open Source projects. Apple has released the source code of the Mac Server OS X, Sun produces the Open Source suite StarOffice in order to contrast Microsoft MS Office, whereas in December 2000 IBM announced its decision to invest in Linux development for promoting the diffusion of its e-business platforms.

These hybrid business models are made possible by the proliferation of licensing agreements that follow the dictates of Open Source. In fact, provided that the fundamental Open Source conditions are respected (full access to the source code, full software usability, freedom of modification and redistribution of software), anyone can elaborate his own Open Source licence. Besides the most famous (and restrictive) GPL under which Linux and over than 80% Open Source programs are released, there are a wide set of licences. Some of them, for instance the BSD (Berkeley Software Development), allow the appropriation of software modifications.

Moreover, other licences, which cannot be considered Open Source given that they protect copyright, provide the release of the source code that particular categories of users are allowed to use. This is the case of the Apple Public Source Licence by Apple and of the Sun Community Source Licence of Sun.

Besides playing an important role in the diffusion process, hybrid models solve simultaneously two problems. On one hand, they secure to potential adopters a wide range of complementary services, which are considered fundamental for reliability and usability of the software, on the other hand they provide monetary incentives for the development of "non sexy" activities, which do not fit the typical motivations of the Open Source community.

Moreover, the existence of commercial subjects using Open Source software guarantees the future survival of the software. The guarantee that software will continue to be produced and developed is a key element in the adoption decision. Users, in fact, can avoid to change applications and platform with high switching costs for infrastructures and personnel training.

The coexistence of proprietary and open software is a possible reading for the success of Open Source movement. The necessity of mixing different remuneration forms with innovative business models belong to the very diffusion dynamic of the free software. Under this point of view, a deep analysis of GPL licence and their above mentioned numerous variants, such as contracts proposed by the new Open Source companies, is a very interesting challenge for economic research..

3. Some figures about Open Source: the empirical relevance of the new paradigm for software production

Several studies (Ghosh, Prakash, 2001; Schimitz 2001) report about 12.000 active Open Source projects. Given that, on average, about twenty programmers work on each project and that each programmer contributes on average to two projects, we obtain a community of

120.000 developers. Such developers have supplied over than 1000 megabyte of code: more than 25 million LOCs[4].

The number of projects posted on the Internet increases continuously. Some of these projects are fated to die but other are getting better and better. Among the most cited (besides the Linux operating system and the Web server Apache) there are BIND, that performs DNS resolution and Sendmail for the exchange of electronic mail. These two programs were the killer applications in their market: no commercial competitor succeeded (or wanted) to attack their supremacy.

The production of Open Source software is characterized by some concentration of the contributions. It is estimated that 10% of the authors write more than 70% of the produced code, with the first 10 developers[5]producing over 25% of the total code. Moreover, the productivity vary across the authors. Only 25 programmers take part to more than 25 projects while the large majority contributes only to a single project. According to Mock et al. (2000), the 15 top developers of Apache added almost 90% of the LOCs and Koch and Schneider (2002) found that, in the GNOME project, 52 programmers out of 301 wrote 80% of the code.

The same structure can be found in the carrying out of the so called “non sexy work”: more than 7.000 individuals provided help on line for the Apache Web server, but 100 of them answered to half of the questions.

Focusing on the diffusion of OSS, we have to notice the exponential growth of the Linux operating system. In the 1998 its market share increased by over than 200%. In the 1999 the profits of Suse, the most important German company for the distribution of Linux, increased by 350%. The explosion of the speculative bubble in the spring of 2000 has slowed down but not stopped this trend: At the same time, the profits of the firms that provide support services for the new kind of software are increasing too: the estimated growth is about 70% per year.

[4] Line Of Code.
[5] Usually Universities and research centres.

Moreover, there are great differences between the server and the client market[6] that are evident by observing both business and public sector markets.

On the client-side Microsoft succeeded in exploiting the *dominant position* gained with MS Dos through the family of Windows[7]. In the server market the situation is quite different, in particular if we focus on web servers. Currently, Netcraft estimates that about 61% of the Internet Web servers run Apache. The penetration rate is quite high also in the case of general-purpose servers (LAN or firm Intranets). Gartner Group (2001) assigns to Linux 10% of the servers in the United State while according to IDC (2001) Torvalds' operating system run on 27% of them[8]. In Europe the public sector adopts Open Source software in 8% of its servers and only 1% of its clients (Schmitz, 2001). The largest diffusion is in Germany and France where many large organisations have adopted Open Source[9]. Focusing only on firms, Forrester (2001) finds that 56% of the 2.500 Top US Companies use some kind of Open Source software, especially on Web servers (33%). Reported figures concerning database-management programs are striking: 42% of them are Open Source while Open Source software in desktop applications is around 1%. The Forrester's study shows very good prospects of development: it estimates that 20% of the present investment expenditure on proprietary licensing will be shifted on services and personnel training by 2004.

A comparable situation was found for Italian SMEs. A survey conducted for Suse Italia by Congenio (2001), shows that 16 SMEs out of 414 use Linux on their servers whereas only one adopt it on the client-side that is dominated by Windows whose share is about 90%. It is very interesting to notice that 80% of the interviewed system administrators admit of being

[6] In particolar in the desktop case.
[7] The Windows 3.1 operating system, which inherited the characteristics of Lisa (Apple) and X Windows (Xerox Parc) in the development of mouse driven graphic interfaces, was released at the beginning of the '90s. The two graphical interfaces for Linux, Kde and Gnome, were released at the end of '90s.
[8] The performances of the Open Source software are particularly good in the field of database server with MySql and PostgreSql.
[9] The French government is very proactive in promoting the diffusion of OSS. In 1999 Open Source solutions were adopted by the Minister of Defence, Education and Finance.

aware of Linux and the Open Source movement in general. In medium-sized enterprises they are 90%.

Stability and security are the main motivations for the use of Linux on the server-side, while Windows is used on the client-side especially because of its easiness of use. Over than 25% of the respondents say that a change of the operative system on the client side is possible and this gives useful indications about the future diffusion of Open Source software. In particular, almost all Linux users on the server-side are in favour of its utilization also on the client-side. Moreover, SMEs seem to be the best candidates for the diffusion of the free software. In fact, the resources used for the licenses could be devolved to the improvement of hardware and to personnel training. This would allow the adoption of advanced ITC solutions by small firms with strong financial constraints.

4. The problem of coordination: software modularity, reusability and the role of the Internet

It is inevitable that an economist, when observing the surprising results achieved by the Open Source movement, should ask: "how to coordinate the efforts of potentially hundreds of developers worldwide?" (Hecker, 1999, 50). At first sight it seems impossible, even though it actually happens. Open Source projects exist and these relatively loosely coordinated software development products have no difficulty in challenging the competition of commercial equivalents. Raymond (1999, 224) notes that before the Open Source revolution, "the whole software design tradition was dominated by Brooks' Law, which stated that a project to which hundreds of developers have contributed can be nothing other than a jumble of unstable, shaky code".

Looking for a possible explanation, some leading members of the Open Source movement went so far to refer to the concept of spontaneous order, comparing this spontaneous and decentralized coordination mechanism to the one lying at the basis of the intersection between demand and supply in markets and of Adam Smith's invisible hand. Clearly this

explanation needs to be integrated with considerations about technology and about the new communication devices that the Internet network makes possible.

About the former aspect, the crucial element is the invention and development of *object oriented programming*. Glass (1999) observes as "object orientation seemed destined to make software development a matter of fastening Lego components together". In object-oriented software, programmes are not a monolithic entity, but can be divided in modules and put together and reused in various ways. Almost all the Linux kernel developers work in a orderly way on kernel subsystems whose files allow the computer to be connected with peripherals or to read video and audio files.

The concepts of modularity and reusability (Bonaccorsi, Rossetto, 1999) thus lie at the basis of software development, where they are rendered even more effective by sharing a common protocol (programming language) in which errors are identified and corrected through the mechanism of compilation.

A clear example of these characteristics is the GREP utility (Generalised Regular Expression Parser) written in C language, which enables identification of a file containing a particular text, basically a more sophisticated version of MS-DOS's Find. GREP is a component of Linux but it was written before the latter appeared and can be used with other operating systems.

According to Linus Torvalds (1999), the key to the success of Open Source is its modularity. Talking about Linux, he argues that "what is needed is as modular a system as possible. The Open Source model of development requires it so as not to have people working on the same points at the same time". And the source of Linux's modularity is its "monolithic kernel".

Coordination is thus made possible by a sensible technical choice that in simple terms involves "keeping the kernel small and limiting to a minimum the number of interfaces and other restrictions to its future development". In this way programmers can add modules without "stepping on each other's toes".

However, even these features do not entirely deal with the issue of coordination, because there is a fundamental piece missing: how do programmers actually communicate with each other? The mushrooming of the Open Source movement, of which Torvalds' operating system is the best example, has coincided with the astonishing diffusion of the Internet network accompanied by the growth of the Internet Service provider industry supplying connections for the general public at a lower and lower prices.

Net is the virtual place where thousands of programmers actually meet, thereby rendering the new development model operative: "once an open source file is posted to the Web, talented programmers world-wide can download it and begin tinkering" (Sanders, 1998, 80). The World Wide Web has become the undisputed realm of the Linux phenomenon, and it has been estimated that the number of Web pages devoted to Torvalds' operating system are "in the neighbourhood of three millions" (Wallich, 1999, 44).

What we are dealing with, then, is a form of coordination that exploits profound innovations:

- in individual tasks, made easily interfaceable by the evolution of software technology;
- in architecture, which has in general become modular due to the evolution of programming languages and in particular due to the strategic choices of Linux;
- in communication costs, which have been slashed and have become insignificant with the advent of Internet.

With these conditions, hierarchic coordination based on ownership assets is not strictly necessary. On the contrary, such coordination would end up depressing the intellectual, aesthetic and pleasure-based motivation that seems intrinsic to the programming community.

5. Open Source software and network externality

The mechanisms underlying the diffusion of Open Source are still not very clear even for its founding fathers. Observing that Linux "it is largely used to control robotic systems and it

has even flown on board the Shuttle" Torvalds (1999) says: "I was aware of the transition from being the sole user of Linux to there being a hundred-odd users, but the leap from a hundred users to millions escaped me".

Accepting Varian and Shapiro's definition according to which "everything that can be digitalized, that is represented by bit sequences, is information" (1999, 3), software is clearly information. From an economic point of view, the production of information involves high fixed costs and negligible marginal costs: the whole mass of cost is concentrated in the first copy, which has extremely low costs of reproduction. On the production side software benefits from economies of scale. The more copies that are produced and sold, the more the high production cost of the first copy will be offset by the low cost of subsequent ones.

From the point of view of demand, the important feature is network externality, that is "demand side economies of scale" (Bensen, Farrell, 1994). This topics give rise to a vast amount of contributions originated by the paper in which Katz and Shapiro gave the classical definition according to which "the utility that a user derives from consumption of the good increases with the number of other agents consuming the good". In the same paper the two authors provide also an exhaustive taxonomy of their sources (Katz and Shapiro, 1985).

Externalities can be direct, indirect or deriving from complementary services. Direct externalities derive from individual physical membership of a network. The most commonly cited example is that of communications networks such as the fax and telephone, in which the utility for potential adopters clearly depends on the number of other families or businesses that are part of corresponding networks. However, there can also be direct network externalities when consumers are nodes of a network that is not physical but virtual or figurative such as in the case of the users of the same operating system.

Indirect network externality characterises the so-called hardware-software paradigm in which two or more complementary goods are joined together forming a system that works only thanks to the contributions of all its components. The positive externality is caused by the fact that "complementary good becomes more quickly economical and available with the growth of

the market for the good compatible with it" (Farrell, Saloner, 1985, 71). This is the case of computers and computer programs, of record and record player, of videotapes and video recorders.

Finally, there is a third source of externality, which occurs with durable goods when "the quality and availability of services related to the good… depend on the number of units of the good that have been sold" (Katz, Shapiro, 1985, 424). One example are the maintenance services provided by car manufacturers – the greater the number of cars a given manufacturer has on the roads, the more reliable and easily available its services will be.

Computer programme users are mainly affected by the first kind of externality: the classic example is the exchange of files. Anyone who has the same programme for running a given file type can read and modify files received from other people who, in turn, can do the same. Hence the importance of sharing a particular type of software – the operating system which runs the whole file system of the computer.

Network externalities influence deeply the diffusion process that, in such a case, corresponds to the problem of the emergence of a standard (Arthur, 1989, 1994, 1996). In these cases a small advantage happening by chance at the very start of the diffusion process is all it takes to favour a standard in such a way that it conquers the whole market that remain locked-in.

The lock-in thesis has recently been the subject of heated debate. Interestingly, the nature of Open Source technologies permits the formulation of a number of points relevant to this debate.

The analysis of standards in economic theory emphasizes how easy it is to fall into the trap well illustrated by Paul David's analysis of the Qwerty- Dvorak battle: "If there are only QWERTY keyboards, typists will study only QWERTY, but if typists study only QWERTY, there will be only QWERTY keyboards". To explain how this comes about, David (1985, 335) refers to the works of Arthur (1989, 1994) about how increasing returns operate in the diffusion process of two technologies that are competing with each other, whereby the more a given technology is used, the greater the probability that it will be in the future. Increasing returns are

an immediate consequence of network externalities, to the extent that Witt (1997) argues that the two can be regarded as synonyms.

Given the presence in the software diffusion process of the previously described direct network externalities, the lock-in mechanism seems an inevitable outcome: if a piece of software manages to gain a significant market share, a virtuous cycle is set in motion such that consumers will have even more incentive to use it, there will be an increase in the supply of complementary products (applications, maintenance) and that particular piece of software will start to dominate the market and become the standard.

As is well known, in Arthur et al. (1983) the concept is formalized using Polya's urn theory. The scheme is that of repeated sampling from an urn containing red and blue balls, where each time a given-coloured ball is extracted another one of the same colour is added. In each period t, the probability of adding red balls is an increasing function of the proportion of red balls in the barrel, and the same applies for the blue balls. The model shows that when the number of balls stretches to infinity, the proportion of balls of the same colour converges in probability to one. Having explained the lock-in, what remains to be explained, however, is what sets it in train. Arthur (1989, 116) refers to the presence of "small events" that "by chance can give one of the two technologies the necessary advantage". If things really are like that, there is no defence against the possibility of inefficient technology establishing its dominance at the expense of another one, which, if adopted, would be superior. This is what happened with QWERTY typewriters, whose performance was inferior to the DSKs of August Dvorak (David, 1985).

The development of Open Source contradicts this prediction in that it is rapidly eroding the market share of Microsoft's dominant standard leading to an equilibrium in which the two paradigms are forced to coexist. In a recent paper, Witt has observed that the mathematical demonstration of lock-in closely depends on a set of initial assumptions. By changing these assumptions it is possible to explain how "newly introduced technology can successfully disseminate in a market despite existing network externality" (Witt, 1997, 753).

Arthur's hypotheses are difficult to defend in the case of software. The author assumes above all that the two competing technologies arrive on the market at the same moment – the "virgin market condition"(Witt, 1997, 762)–, which is clearly unsustainable in the case of software, where the Linux operating system is competing with a decades-old predominance of systems based on and derived from MS-DOS.

In the same way, it is never true that the potential market for a technology is infinite and that the decision to use it cannot be reconsidered. This applies in particular in the case of computer programmes, which have a growing but necessarily limited market (their use is not indispensable to all human activity!) and do not require extremely high fixed capital investments such that the subsequent revision of choices would involve prohibitive costs. Arthur's theory of the independence of individual choices is also hard to sustain.

The economic theory has deeply analysed the influence of the behaviour of other individuals on the adoption choices, starting from the first epidemiological studies, in which diffusion happen for a contagious mechanism from adopters to non adopters (Mansfield, 1961; Bass, 1969, 1980), to the very recent local interaction models[10] in which the decision of every potential adopters are influenced by the ones of a subset of agents considered his neighbours according to a suitable metric. Dalle and Jullien (1999) refer to local interactions to explain the dissemination of the Linux system in place of Windows NT. In their view, what is important in the choice of an operating system is not so much the total number of other individuals who use it but the number of those who do so within the group with whom the individual interacts, i.e. a local network of reference. A different source of diversity emerges in *social heterogeneity* (Manfredi et al., 2000), which is related not to agents' intrinsic characteristics but to the social links they activate in order to exchange information to use in their decision process.

The characteristics of such a network vary according to whether the nodes are represented by Open Source software users or by those using commercial software. A widespread phenomenon amongst the former is in fact the so-called "advocacy" theorized by leading

members of the Open Source movement. This is a form of one-to-one marketing whereby the users of Open Source programmes are invited to convince other members of their group of reference to do likewise and to abandon the commercial sector. Advocacy has an exponential growth: amongst its aims there is that of transforming an individual into a new disciple of the movement and hence into a potential advocate. The role of an Open Source advocate is thus very similar to that of Witt's (1997, 763) diffusion agents who not only disseminate information about the new technology but also try to convince a group of potential users to do so simultaneously so as to contrast the diseconomies which in the presence of network externalities penalize those who adopt the new technology first.

The diffusion process of Open Source software seems then to satisfy the hypotheses at the heart of the existence of Critical Masses in the spread of new technology, which permit an alternation of standards. According to Schelling (1978, 91), the term is borrowed from physics and indicates the quantity of radioactive combustible required to set off nuclear fission. Transposing this concept to the economic and social sphere, one talks of the Critical Mass effect if, when certain variables characterizing a process rise above a given threshold, the phenomenon explodes, so that the system moves from the stable equilibrium in which it was initially positioned and assumes another stable equilibrium. In technology diffusion models, the variable is represented by the number of people who adopt the new technology.

Dalle and Jullien (1999) have carried out computer simulations to analyse the Critical Mass phenomenon – which they call threshold effects– in the diffusion path for Linux. They observe that its emergence depends on the value assumed by a particular parameter α, which is a measure of the "preference of users for standardization" within their network of reference: the greater α is, the more potential users are inclined to adopt the current standard, that is Windows NT. This parameter is therefore also a measure of the proselytising power of Open Source users:

[10] See Fagiolo (1998) for a survey of the literature on local interaction models.

the lower α is, the greater is their strength within each local network. The simulations reveal the presence of threshold effects when the value of α is sufficiently low.

Other authors agree in recognizing the importance in this field of expectations regarding the performance of technology and the final size of the network of users (Huberman, 1998). Analysing the diffusion of two competing technologies whose performance grows as the number of people adopting it grows, Huberman concludes that the waiting period before the achievement of Critical Mass decreases in proportion to the degree of optimism of people's expectations. The transition from the initial equilibrium where everyone uses the old technology to the one where the new technology has replaced the old one also takes place more quickly the more heterogeneous the subjects are. Huberman's simulations demonstrate that if there is a small group of individuals with double the propensity for innovation in relation to the group average, the Critical Mass is reached almost immediately. This once again underlines the role of hackers and their culture in the Open Source movement.

6. Conclusions

The comparison between the proprietary and free paradigm has not necessarily to be done in terms of superiority and of threat to survival.

Starting from the micro-electronic trajectory, the commercial software industry demonstrated to be able to drive the processes of progressive *vertical disintegration* and *autonomization* of the hardware industry (Nelson, Winter, Malerba and Orsenigo, 2001; Torrisi, 2001) and to generate remarkable increases in its productivity together with a reduction in development costs (Cusumano, 1998). The availability of low cost software, both in packages and in customized products, is certainly one of the key elements in the diffusion of ICT in developed countries and in particular in the long growth wave of the second half of the '90s. Then the proprietary paradigm allowed for impressive processes of innovation.

In this sense maintaining that Open Source software brings radically into question the very existence of the proprietary is certainly excessive. The emergence of the free software as a new production paradigm does not necessarily means the end of the proprietary software but the possibility of a new equilibrium in which the two paradigms are going to compete. It seems very difficult to foresee the final outcomes of this dynamic in terms of market share of the two paradigms.

Nevertheless, some elements lead to think that the proprietary paradigm has to converge progressively towards the acceptance of the necessity of making transparent at least part of the sourcing codes, entering into a *hybrid business model*. In fact, some large IT operators (such as IBM) moved toward an explicit legitimisation of the Open Souce movement and the development of solutions that are compatible both with the Microsoft standard and with the Open Source. Moreover many of the users from the industrial and service world (governments and large firms), especially in Europe, began to impose clauses of transparency of the source code in their procurement procedures. We have also to take into account the wave of entry of small software firms that develop Open Source solutions finding their rewards in complementary services. This enlarges the installed base of free software and starts to create network externality effects also in presence of a dominant standard.

The diffusion of the Open Source software had radically different dynamics depending on the presence of the *first mover advantage*. In the sector of client-side operating systems, where Microsoft was the incumbent when Linux appeared, the market share of Window is over 50% on most local markets. On the contrary, in the Web server sector, where the technology became established later and Microsoft had no consolidated advantage, the Open Source system Apache is the incontestable leader. This suggests that who has the *first mover advantage* and quickly aggregates the network externality effect is in the best position for dominating the market. The final competitive equilibrium is going to be placed in some intermediate point between the two monopolies, at a variable distance given the application area and its competition story.

The recent literature that addressed the relationship between path dependence and the nature of the asymptotic equilibrium of diffusion leads to similar conclusions. In the first group of models à la Arthur, in fact, the dynamic equilibrium lead to the *dominance* of a technology on the competing one. The result was clearly very relevant under a theoretical point of view (dominance of inefficient trajectories and lock-in), but scarcely plausible in a number of empirical situations. Nevertheless, some characterisations of the formalization and of the basic assumptions are responsible of this outcome whereas modifications in the models can lead to equilibria where different technologies coexist (Witt, 2000). In brief, coexistence seems a more general situation (Bassanini e Dosi, 2000).

This work shows that the peculiarity of the Open Source movement can be explained using recent developments in the theories of collective action, of coordination in the absence of a central authority and of the diffusion of technologies in the presence of network externality. This has made it possible to throw light not only on how the phenomenon arises and finds a vast economic application, but also on the mechanisms that underlie its increasingly massive diffusion. Many questions do, however, remain open. There undoubtedly needs to be more in-depth analysis of coordination mechanisms, with particular reference to the exact functioning of Open Source projects, in order to understand the role played in them by the project leaders and by the minor figures who do the “non-sexy work”. The search for an explanation of why this “non-sexy work” is done is one of the most interesting challenges that remains for a full economic understanding of the Open Source movement. While both sociology and the economic theory of incentives are able to provide an account of how a Finnish university researcher managed to write the Linux kernel without being paid, it is more difficult to understand what motivates a programmer to work on the realization of a boring graphic interface to make Linux more user-friendly. This is a fundamental point for two reasons: on the one hand, the tendency of Open Source programmes to become more user-friendly will enable their diffusion in increasingly broad bands of the population, on the other, user-friendly programmes and user assistance are the core business of many of the new companies who, by

managing to make profits through Open Source, have demonstrated that in the case of software the word “free” has the meaning of “free speech” and not “free beer”. The mix of commercial and free solutions that goes on through the birth of new companies and the richness of licensing arrangements are other interesting phenomena to be examined in the near future.

The Open Source phenomenon is certainly not the only new information society to throw up interesting economic problems. The transformation of software from private good to collective good realized by Torvalds and the ensuing movement seems to have made Kenneth Arrow’s prediction in the *American Economic Review* paper in 1994 come true, when he warned that in contexts where there is knowledge-intensive innovation, the goods no longer have a fixed nature but are to different degrees both private and appropriable, and public and universal.

The fact that economic theory is best equipped to work on simple cases – with highly specialized theories for private goods (markets) and public goods (public finance) – and struggles with complex cases is obviously a problem. Fortunately, it is not a problem of the real world, which quickly moves on even without economic theory.

References

Arrow K. (1994) *Methodological Individualism and Social Knowledge.* American Economic Review, Economic Review, 84 (2) pages 1-9

Arthur W. B. (1989) *Competing Technologies, Increasing Returns, and Lock-in by Historical Events*. The Economic Journal 99, pages 116-131.

Arthur W. B. (1994) *Increasing Returns and Path Dependence in the Economy*. University of Michigan Press, Ann Arbor.

Arthur W. B. (1996) *Increasing Returns and the New Word of Business*. Harvard Business Review, pages 100-109.

Arthur W. B., Ermeliov Y., Kaniovski Y. (1983) *On Generalised Urn Schemes of Polya Kind.* Cybernetics 19, pages. 61-71.

Bass M. F. (1969) *A New Product Growth Model for Consumer Durables*. Management Science 15(5), pages 215-227.

Bass M. F. (1980) *The Relation between Diffusion Rate, Experience Curves, and Demand Elasticity for Consumer Durable Technological Innovations.* Journal of Business 53(3), pages 551-567.

Bassanini A., Dosi G. (2000) *Heterogenous Agents, Complementaries, and Diffusion. Do Increasing Returns Imply Convergence to International Technological Monopolies?* LEM Working Paper

Bensen S. M., Farrell J. (1994) *Choosing How to Compete: Strategies and Tactics in Standardisation*. Journal of Economics Perspectives 8(2), pages 117-131.

Bessen J. (2001) *Open Source software: free provision of complex public goods.* NBER Working Paper

Bezroukov N., (1999) *Open Source software as a special type of academic research (critique to vulgar Raymondianism). First Monday 4*.

Bonaccorsi A. Rossetto S. (1999) *Modularity in software development. A real option approach,* International Journal of Software Development, Summer.

Chamberlain J. (1974) *Provision of collective goods as a function of group size*. American Political Science Review 68, pages 707-716.

Cogenio s.r.l. (2001) *Indagine circa la propensione delle PMI industriali a migrare verso sistemi operativi alternativi: il caso Linux.* Report presentato a SMAU 2001.

Dalle J. M., Jullien N. (1999) *NT vs. Linux or Some Explanation into Economics of Free Software*. Paper Presented at "Applied Evolutionary Economics", Grenoble, June 7-9.

David P. (1985) *CLIO and the Economics QWERTY*. American Economic Review 75, pages 332-337.

Di Bona C., Ockman S., Stone M., Eds. (1999). *OPENSOURCES. Voci dalla Rivoluzione Open Source.* Apogeo, Milano.

Fagiolo G. (1998) *Spatial Interactions in Dynamic Decentralised Economies.* In Cohendet P., Llerena, P., Stahn, H. and Umbhauer, G. (Eds.), The Economics of Networks: Interaction and Behaviours. Springer Verlag, Berlin.

Farrell J., Saloner G. (1985) *Standardization, Compatibility, and Innovation*. Rand Journal 16, pages 70-83.

Forrester (2001) *Open Source cracks the code.*

Fox J., Guyer M., Humburger H (1975) *Group size and Cooperation.* Journal of Conflict Resolution 19, pp 503-519.

Gartner Group (2001) *An end-user analysis. Linux server market share: where will be in 2001 and how will grow?*

Ghosh R., Prakash V. V.(2001) *The Orbiten Free Software Survey.* First Monday

Glass R. L. (1999) *Of Open Source, Linux and Hype.* IEEE Software 16(1), pages 126-128.

Glass R. L. (2000) *The sociology of Open Source: Of Cults and Cultures*. IEEE Software 17(3), pages 104-IBC.

Green L. G. (1999). *Economics of Open Source Software*. http://www.badtux.org/eric/editorial/economics.html.

Hardin R. (1982) *Collective Action*. John Hopkins University Press, Baltimore.

Hecker F. (1999) *Setting Up Shop: The Business of Open-Source Software.* IEEE Software 16(1), pages 45-51.

Hermann G., Hertel S., Niedner S. (2001) *Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel*. University of Kiel Working Paper

Hertel G. (2002) *Management of virtual teams based on social psycology models.* In E.H. White (Editor), Pabst Publisher, Lengerich.

Hurberman B., Loch C. (1998) *A Punctuated Equilibrium Model of Technology Diffusion.* Xerox Palo Alto Research Centre, Palo Alto.

IDC (2001) Report available at ***http://dailynew.yahoo.com/h/zd/20010611/new_report_questions_linux_server_claims_1.html***

Katz M. L., Shapiro C. (1985) *Network Externalities, Competition, and Compatibility*. American Economic Review 75, pages 424-444.

Koch S., Schneider G. (2002) *Effort, co-operation and co-ordination in an Open Source software project: GNOME.* Information System Journal 12, pages 27-42.

Lakani K., von Hippel E. (2000) *How Open Source works: "Free" user-to-user assistance.* MIT Sloan School of Management Working Paper #4417.

Lerner J., Tirole J. (2001) *The Open Source movement: key research questions*. European Economic Review 45, pages 819-826.

Liebovitch E. (1999) *The Business Case for Linux*. IEEE Software 16(1), pages 40-44.

Liebowitz S. J., Margolis S. E. (1990)*The Fable of Keys.* Journal of Law and Economics 33, pages 1-26.

Liebowitz S. J., Margolis S. E. (1996) *Market Process and Selection of Standards.* Harward Journal of Law and Technology 9, pages. 283-318.

Malerba F., Nelson R., Orsenigo L., Winter S. (2001*) History-Friendly models: An overview of the case of Computer Industry.* Journal of Artificial Societies and Social Simulation 4(3)

Manfredi P., Bonaccorsi A., Secchi A. (2000). *Heterogeneity in New Product Diffusion.* LEM Working Paper.

Mansfield E. (1961) *Technical Change and the Rate of Imitation*. Econometrica 29, pages 741-766.

Marwell G., Oliver P., Prahal R. (1993) *The Critical Mass in Collective Action. A Micro-social Theory*. Cambridge University Press, Cambridge.

Mauss M. (1959) *The Gift. The form and the reason for exchange in archaic societies.* Routledge, London

Mockus A., Fielding R., Herbsleb J. (2000) *A case study of Open Source software development: the Apache server.* In Proceedings of the 22nd International Conference on Software Engineering, pages 263-272.

Olson M.(1965) *The Logic of Collective Action.* Cambridge University Press, Cambridge Massachusetts.

Perkins G. (1999) *Culture Clash and the Road of Word Dominance*. IEEE Software 16(1) pages 80-84.

Raymond E. S. (1999) *The Cathedral and the Bazaar.* http://www.redhat.com/redhat/cathedral-bazar/.

Roger E.M. (1995) *Diffusion of innovations*. Free Press, New York, 4th edition.

Rossi C. (1998) *Il Problema dell'Azione Collettiva nella Teoria Economica: Una Rassegna di Alcuni Recenti Contributi*. Degree Thesis. Economics School, University of Pisa, Italy.

Sanders J. (1998) *Linux, Open Source and Software's Future*. IEEE Software 15(5), pages 88-91.

Schelling T. (1978) *Micromotives and macro beavhior*. New York, Norton

Schimitz P.E. (2001) *Study into the use of Open Source software in the public sector*. An IDA Study (Interchange of Data between Administrations).

Stone M. (1998) *Science of the New Renaissance*. http://www.edventure.com/release1/1198.html.

Torrisi S. (1998) *An international study of software industry.* Edward Elgar Publishing

Torvalds L. (1999) *Il vantaggio di Linux.* In Open Sources: Voci dalla Rivoluzione Open Source, a cura di Chris Di Bona, Sam Ockman, Mark Stone. Apogeo, Milano.

Ullman E. (1998) *The Dumbing Down of Programming.* 21st Salon, 13 May, http://www.salonmagazine.com.

Varian H. L., Shapiro C. *Information Rules.* ETAS Libri, Milano.

Wallich P. (1999) *Cyber View The Best Things in the Cyberspace Are Free.* Scientific American March, page 44.

Witt U. (1997) *Lock-in vs. Critical Masses. Industrial Changes under Network Externalities*. International Journal of Industrial Organisation, 15 pages 753-772.

Witt U. (2000) *Path-dependence in Institutional Change*. In: K. Dopfer (ed.), The Evolutionary Principles of Economics. Cambridge, Cambridge University Press