The evolution of free/libre open source software.

Introduction

The *Free Libre Open Source Software* represents an outstanding example of *"open development model of technological knowledge"*. It has been studied in several researches that produced valuable illustrations of the way it works. Our understanding of its principal features is growing exponentially and an entire new literature on open source has been created. However there appears to be an important gap in the literature: *the origin of the phenomenon*.
The following chapter attempts to tackle this issue by analyzing the long-term technological history of the Free Open Source Software; the main research questions at stake are: "Is the phenomenon completely new? and if it is not totally new, where does it come from?" and, more generally, "how did open source software developed over time?". As a consequence the present work focuses primarily on the analysis of the free/open source software *history of technological change* over a period of almost sixty years. I adopted a multidisciplinary approach to analyse the network of relations emerging between inventions and technological innovations, as well as economic determinants and intellectual property rights regimes throughout the period considered. Therefore, I attempted to investigate the origins of the phenomenon as a way of understanding its evolution.

The central aim of this chapter is to examine the technological history of the Open Source phenomenon[1]. The following work will explore a specific portion of the history of modern computing (Ceruzzi, 2003) and the history of software industry in order to unfold the historical evolution of free/open source software or more precisely, the "path dependence evolution" (Antonelli, 2005) of the open model of knowledge management that characterizes the phenomenon. The analysis will show that the so-called free/open source cannot be considered a new way of developing software programs as it is rooted in the way computer scientists have always faced the problem of making communication between man and machines easier and more efficient, consequentially it would be rather considered as the result of a long path of growth in knowledge management capability. Thus the following history will offer some evidences and arguments on the tight relationship between the evolution of modern computing, more precisely the evolution of a small but relevant part of this history, and the improvement of communication technologies and knowledge management techniques. Following this approach, technological communication is likely to influence the way technological understanding develops. It basically shapes the growth of knowledge and the

[1]Lorenzo Benussi - **lorenzo.benussi@unito.it**
Laboratorio di economia dell'innovazione "Franco Momigliano", Department of Economics, Univeristy of Turin.

I choose to use the expression "Open Source Phenomenon" in order to indicate the complex milieu of technologies, institutions and economic factors that characterize both the Free Software and the Open Source Software movement. The expression aims at grouping these two different philosophies together in order to make easy the analysis of their common features and the comparison with the proprietary software model.

following example will provide useful evidences in this direction.

First of all it is worth specifying why I assume communication is so important as the determinant of change in this particular case and what I mean exactly by communication. The term acquires a specific meaning since I am dealing with Computer Software, based on some sort of programming language. In fact, the software is a technological artefact that retains all the characteristics of a communicational medium, though a highly technological one, addressed to make communication possible between human beings and computers. More precisely it is a way to ask proper questions to machines in order to perform several operations efficiently. By the same token, a programming language can be seen as a way of codifying information that makes interaction between man and machines easier. As a consequence, the software is a medium to perform communication but it is also a way to codify problems and express them in a machine readable way: it is a kind of language. As every language it has rules and exceptions, it can be misinterpreted, but it still maintains a consistent general order and logic, it retains a stable form. However, as technology, it necessarily also changes over time in relation to a complex set of variables, namely economic, legal, social and technological ones. We can roughly say that as software code and programming language evolve so do the technology and the technological knowledge. Hence I chose to set up my historical examination of the FLOSS evolution on this dialogic path of change in order to understand how its technological language has been created, what sort of particular questions there are to solve, which internal logic governs it and how it has reacted to the natural evolution of modern computing.

In order to accomplish the task, I have chosen to collect a multidisciplinary dataset about the technology that I assume characterizes the FLOSS as I needed to create a classification of its inner features. Even if it seems natural to focus on the operating system technology, more precisely the GNU/Linux operating system technology, only its examination was not enough because, as I said earlier, the technology is only one part, or better one portion of the explanation I am looking for. So I took into account four different dimensions of change affecting this history: the progress in technology, the growth of the software industry and the computer market, the evolution of intellectual property rights on software programs and the mutable relationship between computer technology and society at large, or a part of it. Once I set these four directions of research, I began to collect data according to them. I started from the present debate on free/open source and I went back to the beginning of the phenomenon. The genealogical path I followed was quite clear: Gnu/Linux derives from Unix, which comes from technical ideas developed around time-sharing operating systems, as the MIT MULTICS system. Moreover the FLOSS community is built on the Internet, which can be considered a grandchild of ARPANET. Fortunately the two streams come from the same "great expectations" on the possibilities of computer technologies: the enhancement of human problem-solving capability by the perfection of coding, transmitting and managing information that was expressed for the first time in 1945 by Vannevar Bush, which some years later founded

the National Science Foundation, and perfected in 1960 by J.C.R. Licklider, which, after being trained in the SAGE project, contributed as main character in the development of ARRPAnet. What they tried to obtain was a man-computer natural interaction by Symbiosis[2], namely they wanted to improve the communication capability of machines. So communication is important twice: firstly because it is the main technological goal of the FLOSS and secondly because it is the key material FLOSS is made of. Summing up, the work aims at better understanding the complex relation between innovation, knowledge and communication taking into account a single but meaningful case study: the free/open source software.

This chapter is divided in three sections. The first and main one presents the analysis of the FLOSS history; the second tries to sum up the lessons learned during the investigation and the last one points out some methodological problems I faced during this investigation.

Analytical review of free/open source software history

The idea of free software may seem a bit eccentric: the collective ownership of intellectual property rights is surprising and many ideas referred to the so-called "Linus's law" and the "Linus's philosophy"[3] may appear odd and naive, particularly for economists that usually analyse profit-maximising agents. In spite of its queerness or probably because of this the academic community has shown a widespread attention to the open source phenomenon and the amount of academic papers, researches and surveys on this subject is increasing day by day[4].

One of the common issues of the debate is the attempt to organize the notion of open source software under a well established analytical framework[5]. The whole system may seem out of the ordinary because of the noticeable "free of charge" way of "trading" information. As a matter of fact everybody can download free/open source software, he can look at how it works, use it as well as sell it. Although the process may appear pointless, it has been working for almost twenty years and the trend is growing remarkably. But how is this possible? The

[2] As the author points out: *"Man-computer symbiosis is an expected development in cooperative interaction between men and electronic computers. It will involve very close coupling between the human and the electronic members of the partnership. The main aims are 1) to let computers facilitate formulative thinking as they now facilitate the solution of formulated problems, and 2) to enable men and computers to cooperate in making decisions and controlling complex situations without inflexible dependence on predetermined programs."* Licklider, Man-Computer Symbiosis, 1960

[3] Linus's law states: "*given enough eyeballs, all bugs are shallow*". More formally: "Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone." The rule was written by Eric Raymond in his book the Cathedral and the Bazaar. Linus Torvalds also explains this idea, as Linus's Law, in the prologue to the book The Hacker Ethic (Himanen, 2004). He says: *"all of our motivations fall into three basic categories. More important, progress is about going through those very same things as "phases" in a process of evolution, a matter of passing from one category to the next. The categories, in order, are "survival", "social life", and "entertainment"*. The law is a metaphor of the *open source technological philosophy* of software development, personified by the enthusiastic programmer, which finds entertainment as the most valuable motivation. The two concepts are partially different: the former is centred on the effects that the open development model should have on the developing process, the letter is about the inducements that lead the developer's structure of choices.

[4] The number of researches on Free/Open Source Software is remarkably rising. One may see the repository of the MIT Free/Open Source Research Community (http://opensource.mit.edu) in order to appreciate the vast literature.

[5] An accurate overview of the different interests that the topic provokes, for example in the field of Innovation Studies and Economics of Innovation can be found on the "Special Issue on Open Source Software", Research Policy, Volume 32, Issue 7, July 2003.

answer is not easy to find as the phenomenon is likely to have all the characteristics of a complex technological phenomenon strictly related to: 1) the history of modern computing, 2) the upward trend of information and communication technologies, 3) the pervasive use of digital media and 4) digital network.

Nowadays the FLOSS is widespread in many sectors of the so-called ICT market, from server applications to embedded devices, but the cause of its success is still quite undefined. Almost certainly it is the mix of a complex set of technological items, a "way of technological development" more that a "technology stricto-sensu", that has been influenced not only by technological but also by economics, institutional and social factors. Thus the problem still probably needs an appropriate explanation.

I am of the opinion that it is worthwhile beginning with the examination of the expressions *free software* and *open source software* and that the proper comprehension of these terms requires a historical review of the way modern computing has been developed. In so doing I will take into account several aspects that may be useful to map the sequence of events that lead to the free/open source software movement creation and development; each one has significant implications on and contributes to FLOSS as it is today. Furthermore this particular overview is meant to shed light on the roots of the distinctive *way of thinking* that supports the FLOSS movement.

Although the relation between historical facts is by definition not linear, remarkably dealing with complex topics as scientific discoveries and technological changes, I choose to start from the *models of processing information*, developed during the second half of the previous century, and the *communication technologies* that followed those ideas because, as I am going to show later, both have affected the development of FLOSS. Following this unit of analysis I divided my overview in stages that set up the evolutionary path leading to today's FLOSS phenomenon. The division is designed to show how the idiosyncratic features of the FLOSS technological knowledge and communication model emerged. Consequently the following review attempts to identify and to describe the origin of the *technological paradigm* (Dosi, 1982) that characterizes the FLOSS phenomenon and the *technological trajectories* emerged within its path dependence process of change.

I decided to divide the history of open source software, or better the history of the technological change that characterizes the phenomenon, in sequential chronological phases. Although it might be questionable to compel the natural stream of technological history, it is a matter of fact that every interpretation of a phenomenon, whether economic, technological or sociological, needs *historical map*. Therefore this study tries to accomplish such a challenging task.

I think it would be a good starting point to stress the strong link between several technological breakthroughs in communication technologies that date back at least thirty years and the Open Source phenomenon, though it has remarkably changed during the period according to transformations of computer communication capability. Even though it would be

straining to recognize the open source software since the beginning of this history, some features, at least its high-level architectural ones, have appeared since the beginning of it. In fact, the core ideas substaining the open source phenomenon are mainly based on a typical way of seeing the development of computer machine and consequentially computer software that intends to improve the human power over the most complex human activity: "cognition".

Following this research idea, computers have the characteristics of a technological system addressed to manage information and data with the clear aim of increasing the knowledge management capability of human beings. Hence the central "genetic" attribute of this system is the commitment of managing knowledge in the most efficient way and eventually creating new knowledge: the system as a whole might be defined as a General Purpose communication technology. The following history will focus on such DNA[6] and it will attempt to answer where it comes from and how it has interacted with different environments during its development. The examination will show us how the development of practices and experience using a specific *network of communication technologies* influenced, or better, shaped the path dependence process of FLOSS evolution.

The six steps of FLOSS history

The following overview is divided in six periods that represent six steps of the FLOSS evolution. Each step is the origin of the next one and it is linked with it by both technological and economic causation. Each stage begins with one representative fact, which sums up the major features of whole period and represents the enhancing mechanism that distinguish it as well as the main variation from the previous one. The point of this division is creating an analytical interpretation by composing a chronological map of the FLOSS development in order to better comprehend its past and eventually its future. Summing up, the following division aims at organizing the major concepts the open source software is based on taking into account economic as well as technological factors.

The first stage describes the rise of a new era that influenced the following system of innovations because the basic questions about the development of computer technologies were formalized at that time. It starts after the Second World War with the seminal contribution of Vannever Bush, and it covers the first 20 years of modern computing development. New scientific and technological challenges characterized this period, as well as the creation of a new industry based on computer machines, and its legacy still effects the software industry today. The creation of the UNIVAC machine and the SAGE system represent the foremost projects of this period that paved the way of future innovation as long as defining the parading for the following development of modern computing are concerned.

[6] The idea of technological DNA comes from an Evolutionary prospective on Technology and Innovation. This approach has proved to be productive in several disciplines, from Biology to Linguistics, and I think it is an inspiring metaphor for my history.

The second phase focuses on the foundation and the evolution of one of the chief projects of modern computing: The Project Mac at the MIT, which created the influential Compatible Time Sharing operating System CTSS and the Multics System.

The third period analyses the birth of the language the open source is based on: UNIX. It describes the development of UNIX operating system from its research-oriented "origins" to its "maturity" witnessed by the creation of Sun Microsystems, one of the most influential and innovative Unix-based and open source oriented company.

The fourth step analyses the creation of the Personal Computer technological system and the outburst of the related market. New companies began to produce "personal computer" as MITS Altair 8800 and the Apple Macintosh as well as new software companies, as Microsoft, entered the market. Suddenly the new system, composed by the PC and its proprietary programs, becomes the most valuable products of the ICT industry.

The fifth segment takes into account the institution of the GNU Project and the Free Software Foundation. Moreover during this period Internet widespread and the World Wide Web were invented. Meanwhile the Open Source Initiative was founded in order to support the growing technical, economic and political success of free/open source movement.

Finally the sixth period describes the origin of Gnu/Linux and the changes that its new approach had on the open source movement. The period starts in 1991 when Mr Torsvald posted on-line the first address to freely download his Kernel Program and it finishes in 2001 with the advent of the so-called WEB 2.0.

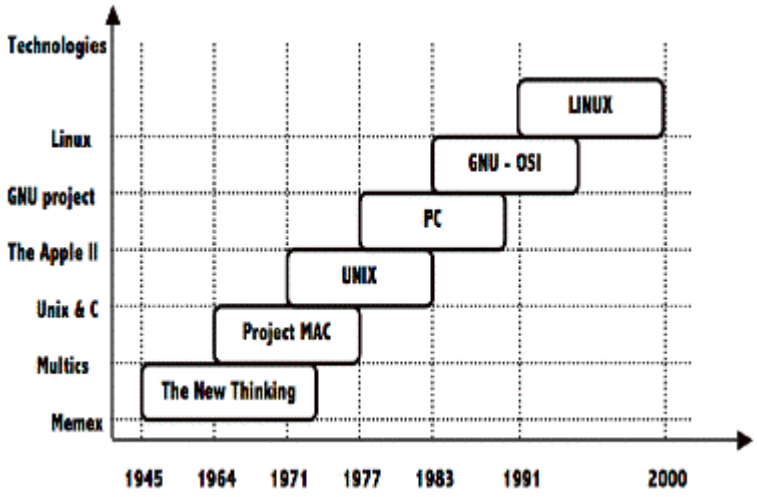

Figure 1: the six steps of FLOSS history

1. The "new way of thinking": 1945 – 1969

The genesis of the phenomenon origins from the seminal contribution of Vannevar Bush[7]: the

[7] I choose to tribute such consideration to Vannever Bush because he played a chief character in the post World War II scientific and technological environment in US. But he also made eccentric previsions. See Wikipedia: *"Vannevar Bush has an unfortunate eponym: vannevar owing to his habit of overestimating technological challenges. He asserted that a nuclear weapon could not be made small enough to fit in the nose of a missile as in an intercontinental ballistic missile. He also predicted "electronic brains" the size of the Empire State Building with a Niagara Falls-scale cooling system, although this could be considered a metaphor for systems such as Google's entire collection of Linux servers, whose collective size and thermal*

idea of "memex" machine. The end of this period is set in 1968 with the creation of ARPANET.

Main features	The Memex machine, the National Science Foundation, the topology of a distributed telephone networks and the project of ARPANET.
Chief ideas	The "memex" machine(Bush, 1945), information entropy (Shannon, 1948), the Distributed Network (Bran 1964), the "Galactic Network" concept and the computer as communication device (Licklider, 1960 and 1962).
Main problem	To communicate and to mange an increasing amounts of data in the most suitable way.
Link with the successor	The commitment of improving technological communication capability.
Permanent results	The creation of main institutions (NSF) and technological network (ARPANET-Internet).

2. The "big" MAC: 1963 – 1975

The second period is characterized by the creation of the Project Mac at the MIT, probably one of the most influential projects in the entire history of modern computing, It finishes with the election of Michael Leonidas Dertouzos as the fourth director of it and the consequent change of the name in LCS: Laboratory of Computer Science in 1974.

Main features	Time-Sharing System, On-Line System.
Chief idea	Time-sharing system (Fano, R. M. and F. J. Corbato, 1966), Hypertext (Nelson 1965), The *bug*: today knows as *PC mouse* (Engelbart, 1968).
Main problem	To create a modular system able to run communication among networks of computers, to project the computer as personal assistant for intellectual workers.
Link with the successor	The need to perfect the time-sharing system and evolve the Graphical User Interface.
Permanent results	The creation of a computer network as a public utility, the first experiments of the Graphical User Interface, the conceptualization of hypertext.

3. The language: 1971 – 1982

The third period is focused on the creation of UNIX the fundamental operating system of which GNU/Linux can be seen as technological "dialect" or "clone". It starts with the first conception of UNIX by Ken Thompson and Dennis Ritchie at the AT&T's Bell Telephone Laboratories and it finishes with the foundation of SUN Microsystems in 1982.

Main features	The UNIX system, the C programming language.
Chief ideas	The concept of "pipe" (Thompson and Ritchie 1975), the hacker way of sharing technological information and the "art" of tinkering.
Main problem	Building a simple but affective operating system able to run on

emissions may very well be on the same scale." (http://en.wikipedia.org/wiki/Vannevar_Bush)

	different hardware platforms.
Link with the successor	The UNIX system is the basis of GNU/Linux as well as many other system both open source, as FreeBSD, or proprietary, as UNIX System V Release 5.
Permanent results	The style of programming (KISS), the need for modularity and portability in operating system design, the creation of new networked computers (BSD plus TCP/IP).

4. The birth of PC: 1977 – 1991

The fourth period briefly describes the birth of the Personal Computer and it finishes with the publication by Tim Barners Lee, on August 6, 1991 of a short summary of the World Wide Web project on the alt.hypertex newsgroup.

Main features	The personal computer, the GUI, spreadsheet programs, multimedia programs, the WWW.
Chief ideas	The "plug and play" personal computer already packaged and ready to use, the growing importance of software and the central role of network computing.
Main problem	How to create the most desirable PC and how to connect it efficiently with the digital network.
Link with the successor	The upward diffusion of the proprietary software model, actually motivated the birth of FLOSS.
Permanent results	The PC, the WWW, the diffusion of Digital Media.

5. The GNU Project and the Open Source Initiative: 1983 – 1998

The fifth phase is centred on the origin of the GNU project, the Free Software Foundation and the Open Source Initiative. It starts September 27, 1983 with the public announcement of the creation of the GNU project by Richard M. Stallman on the net.unix-wizards and net.usoft newsgroup. It finishes with the foundation of the Open Source Initiative by Bruce Parens and Eric Raymond in 1998.

Main features	The creation of a new operating system similar but not UNIX and the creation of tools to manage a vast community of developers that work together freely for its development.
Chief ideas	The concept of "copyleft" and the creation of the General Public Licence (Stallman & Moglen 1985). The foundation of the open source definition (Parens at al.,1998) and the definition of the differences between "the cathedral and the bazaar" model of software development (Raymond, 1997).
Main problem	How to keep the software free, how to support a large development community and how to expand the movement.
Link with the successor	The GNU project become a fundamental part of the GNU/Linux system, the Open Source Initiative promotes the diffusion of GNU/Linux providing organizational capability.
Permanent results	The practical demonstration that a community of people can

	collaborate with no compensation in order to produce very good software, moreover the creation of a set of "institutional tools" as specific licences able to protect it.

6. Linux age: 1991 - present

The Linux age starts on the 5th of October 1991 at 8:53 am when Linus Benedicts Torsvald publishes the announcement on comp.on.minix newsgroup with this heading: "free minix-like kernel source for 386-AT". It can be seen as the first public appearance of LINUX. Linus wrote: *"The Sources for this pet project of mine can be found at nic.funet.fi (128.214.6.100) in the directory /pub/OS/Linux."*[8]. This period ends in 2001 with the success of Google and the advent of what Tim O'Reilly defined the WEB 2.0.

Main features	The Linux kernel.
Chief ideas	Combining the power of Internet with the power of the free software community; the Linus's philosophy.
Main problem	How to use the new medium better, the WWW in order to develop better software.
Permanent results	The "Linux uprising".

Step One. The "new way of thinking": 1945 - 1969

The genesis of the intellectual atmosphere that engendered the practice of sharing software programs is linked to the new collaboration practices developed by researchers based in a number of laboratories and private research centres in EU and US after the Second World War. Moreover the idea of sharing experiences and practices between computer users has always characterized the development of computers even on the more commercial and industrial side of the computer market, as the project SHARE clearly demonstrates. Following this genetic attribute of modern computing, the particular story of the birth of what is nowadays called open source software is strictly related to the tradition of *knowledge interaction* typical of the early years of computing. The history is vast and it is not the purpose of this overview to illustrate it in details, though it is useful to examine some factors, related to the development of the concept of "coding and transmitting information" that may give us precious insights of where the FLOSS phenomenon comes from.

The particular history of open source, or should I say the history of the FLOSS "way of thinking", starts after the Second World War. In fact during the War many scientists, like the physicist community during the Manhattan Project, were forced to collaborate in order to

[8] Linus reported that the name LINUX was not predetermined and it comes from the server's directory name where it was available for the first time: /pub/FLOSS/Linux.

create practical devices, and possibly weapons, useful for the whole of society. The idea was relatively new and it represented an extreme attempt to face the drama of the War with scientific instruments. As everybody knows that attempt succeeded in the most terrific way human beings have ever seen. Apart from its "remarkable" practical effects, which still influence scientific thought, that experience has created what may be called a strong consciousness that the history of science had changed and that a new policy of science had to be signed.

The growing consciousness that the supremacy in technology leads to supremacy in politics was reinforced by the success of the Manhattan Project and consequentially it created a huge stream of funds for research activities and a number of fundamental projects started. In fact the viability of financial resources suggested new possibilities for scientists and new promises regarding the potential of technological progress started to take form. Consequentially after the War politicians and scientists spent a notable effort in order to understand what would be the future of their society and what would be the most suitable technology for this process of change. The attempt generated two particular reactions that are relevant to our story: a) the common understanding of the importance of Science for Society and b) the challenge of producing, communicating and managing larger and larger amounts of data. These two ideas affected *the way the scientific community reflected on its future* and the efforts they decided to spend in order to achieve common technological objectives.

One of the foremost challenges was the *perfection of communication technologies* in order to get over the problem of processing data and the solution was sharing energy to overwhelm the difficulties of new complex calculations. Therefore the concept of sharing technological resources may be viewed as *the birth of a new kind of development process* within the scientific communities[9]. Even though it was only one part of a general revolution, which occurred after the drama of the War, it seems appropriate as the *symbolic start* of the new way of technological development, investigated here. Moreover this diffused scientific and technological endeavour had another cardinal effect on the scientific community that could be named "connected-community effect". It is related to the fact that scientists lived for a long period of time in the same place in order to interact with each other and to share ideas with the explicit aim of solving military problems. This *experience of interaction* required to be continued after the War. While military investments increased, new ideas of cooperation spread. In particular it seemed to be rational to share resources to reach bigger power of calculus and accuracy of analysis. *A new way of storing and organizing information ought to be created and the scientific community was likely to be ready to accept the challenge.*

The concept of hypertext was the first answer to the need of "thinking connected". The new way of thinking widespread among major universities and research laboratories, as well as private companies, and it raised resources to a stable point: creating better communication technologies. On the same path, after some years, another challenge was formalized: the idea of

[9] See: Bush, V. (1945) *Science - The Endless Frontier*, Washington, D.C., United States Government Printing Office

connecting computers in a way that can be preserved from enemy attacks. That new idea of communicational infrastructure began to influence the vision of people involved in paving the way of innovation. The plan was basically to enable a distributed, mashs-like network (Baran, 1964), instead of the current centralized system used by that time. This topological design influenced all the future development of communication technologies and generated a stream of new possibilities. Major Research Centres gained the opportunity to connect the computers hosted in their own laboratories with external data-centres and consequentially enhance their ability in running growing flows of information.

Furthermore, as I will show in the next paragraph, the history of *tinkering with computer programming* began in the sixties. After the Massachusetts Institute of Technology had bought the first PDP-1[10] computer, produced by Digital Equipment Corporation (DEC – Digital)[11] in 1961, researchers inaugurated the practice of creating a community around the new powerful machines. Then they started to develop a sub-culture of computer programming which can be seen as the ancestor of the so-called "hacker culture" of today. That culture was based on the belief that open technical communication is the key to fostering the rate of innovation. Few years later, in 1969, the ARPAnet project started. The development of a new communicational infrastructure involved a great number of intellectual forces and it induced to try new communication opportunities. The same year Unix was born out of the mind of a computer scientist at AT&T's Bell Laboratories, Ken Thompson. The original Unix can be considered as a grandchild of the Compatible Time-Sharing System (CTSS)[12], one of the first "time-sharing"[13]

[10] PDP is an abbreviation for Programmer Data Processor, the name of a series of computers, several of them pioneering and very influential, made by Digital Equipment Corporation. They have that name because at the time of their launch, computers were large and expensive machines, and the PDP technology aimed at a market that couldn't afford the larger computers. The various PDP machines can generally be grouped into families based on word length. With the notable exception of the 16-bit PDP-11, the architectures show strong similarities with the 36-bit PDP-6 and PDP-10 architecture being the most elaborate.

[11] DEC was one of the most pioneering companies in the early years of computers. The acronym was once officially used by DEC but discarded later in favour of "Digital" in order to avoid trademark disputes. Compaq, which subsequently merged with HP, acquired Dec in 1998; in 2004 their product lines were still produced under the HP name. Ken Olsen, a Massachusetts engineer who had been working at MIT Lincoln Laboratory on the TX-2 project, founded the company in 1957. The TX-2 was a highly innovative project committed to create a transistor-based computer and when that project ran into difficulties, Olsen left the MIT and founded DEC with Harlan Anderson, a colleague from MIT. In 1961 the company started the construction of their first computer: the PDP-1.

[12] CTSS is one of the first time-sharing operating systems; it was created within the Project Mac at MIT. CTSS was first published as well as operated in a time-sharing environment in 1961; in addition, it was the system with the first computerized text formatting utility and one of the very first to have inter-user electronic mail.

[13] Time-sharing is an approach to interactive computing in which a single machine is used to provide apparently simultaneous interactive general-purpose computing to multiple users by sharing processor time. It is based on the idea that multiple users could share a machine by using one user's idle time to service other users. Similarly, small slices of time spent waiting for disk, tape, or network input could be granted to other users. Bob Bemer first described the concept in early 1957. The first project to implement a time-sharing system was initiated by John McCarthy in 1957 and the results of it, the CTSS Operative system, appeared in November, 1961. As the director of Project Mac Fernando J. Corbatò pointed out: *"The basic technique for a time-sharing system is to have many persons simultaneously using the computer through typewriter consoles with a time-sharing supervisor program sequentially running each user program in a short burst or quantum of computation. This sequence, which in the most straightforward case is a simple round-robin, should occur often enough so that each user program which is kept in the high-speed memory is run for a quantum at least once during each approximate human reaction time (~.2 seconds). In this way, each user sees a computer fully responsive to even single key strokes each of which may require only trivial computation; in the non-trivial cases, the user sees a gradual reduction of the response time which is proportional to the complexity of the response calculation, the slowness of the computer, and the total number of active users. It should be clear, however, that if there are n users actively requesting service at one time, each user will only see on the average 1/n of the effective computer speed. During the period of high interaction rates while debugging programs, this should not be a hindrance since ordinarily the required amount of computation needed for each debugging computer response is small compared to the ultimate production need".*An experimental time-sharing system, Corbató, Daggett, Daley (IFIPS, 1962).

operating systems, developed within the pioneering Multics project. Multics, even if it was a technological breakthrough but a commercial dramatic failure, attempted to create a feature-packed "information utility" that would gracefully support interactive time-sharing of mainframe computers used by large communities of users. Unix is likely to be a successfull attempt to patch the problems unveiled by the Multics project and to improve its achievements. In particular the key point was to improve the possibility suggested by the *time-sharing way of thinking*. Dennis Ritchie, which is well known as the co-inventor of Unix and the inventor of the C programming language, was the first collaborator of Thompson. Both were involved in developing a computer system that should have been innovative and reliable following the technological trajectory of time-sharing system. As Dennis Ritchie pointed out[14], they had had a specific idea of developing an open technical environment since the end of 60s.

Together the design of ARPAnet and the creation of Unix should be seen as the foundation of an entire new way of thinking about computers. The theme of computers machines not just as logical devices but also as hubs of a community was in the air. Two of the main components of free and open source software phenomenon start their history: *the communication network and the software language.*

Step two. The "big" MAC: 1964 – 1975

The birth of a technological paradigm can be analysed as the sum of attempts to solve a technological problem. The progressive gathering of trials and errors shapes the way of conceptualising solutions about a technological problem by focusing on a single method, which looks more efficient than others. In the case of the system of software the open source software belongs to, this way of thinking can be - at least to some extent - assigned to the concept of "time-sharing" and in general of *sharing the capacity of processing data*. As a matter of fact the sequence of attempts to solve this problem created a number of scientific insights and fundamental technological solutions that influenced the future development of software technologies. In particular the idea of creating a public utility of computers started to be a central issue.

I decided to investigate the story of time-sharing systems following the story of the MIT Project MAC, which generated in turn the MIT Artificial Intelligence Laboratory and the MIT Laboratory for Computer Science, because the sequence of remarkable inventions created by this project constitutes the DNA of FLOSS. As a matter of fact Richard Stallman, who founded

[14] *" What we wanted to preserve was not just a good environment in which to do programming, but a system around which a fellowship could form. We know from experience that the essence of communal computing, as supplied by remote-access, time-shared machines, is not just to type program into a terminal instead of a keypunch, but to encourage close communication."* Dennis M. Ritchie, "The Evolution of the UNIX Time-sharing System", AT&T Bell Labs Technical Journal, vol. 63 no. 8, part 2, October 1984, p. 1578.

the Free Software movement, started his career with this project. By the same token one of the major features of its typical model of knowledge development was the commitment of creating communities and particularly the creation of an epistemic community (Antonelli, 2005) enhanced by a technological network, which was not only a technological challenge but also *the "method" to perform research* and experiments of this project. Therefore it can be viewed as the historical foundation of the technological trajectory I'm enquiring on here.

As John McCarthy, who started working on time-sharing since his first arrival at MIT in 1957, said at the MIT Centennial in 1961: *"If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility.... The computer utility could become the basis of a new and important industry."* But by 1955 most computers were big, complex and expensive machines built for few large corporations, government agencies, and big universities. Fernando J. Corbató, who was associate director of MIT's Computation Centre during that time, and his colleagues were convinced that one of the central problems of these "dinosaurs" was the limitations to run on the machine only one program at the same time. It means that if many people were waiting to use the machine, it would take a very long time before one would get the result. But they understood that there was a way to get over this problem and they called this technique *time-sharing*. Instead of devoting the computer power to one task at the same time, a time-sharing system would work on many tasks simultaneously, switching among different jobs so fast that each user would have the illusion of real-time, interactive use of the machine. John McCarthy first conceived interactive time-sharing. The development started in 1959 when McCarthy and his colleagues, including Steve Russel and Arnold Siegel, had the permission to modify MIT's IBM 704 in order to make a elementary interrupt mechanism. By 1960 IBM had been persuaded to modify the Computation Center's 7090 for time-sharing. In 1961, a rudimentary prototype with four terminals was running on the IBM 709. During the spring of 1962 the system switched to the Computation Center's IBM 7090 and the time-sharing system had achieved the hardware basis required to demonstrate its effectiveness, a remarkable success for the time-sharing ideas.

The second building block fell into place in the fall of 1962 when J.C.R. Licklider[15] was recruited away from BBN to a two-year period at the U.S. Department of Defense's Advanced Research Projects Agency (ARPA), where he became the founding director of ARPA's Information Processing Techniques Office (IPTO). Licklider, as experimental psychologist, was excited by the potential of interactive computing and he was determined to explore its possibilities. Consequentially he spent the first year of his new job travelling to the country's major research facilities, trying to coax them into action. One of his first stops was at the MIT.

[15] J.C.R. Liklider is one of the main characters that contributed to the development of the Internet's technological architecture. He was one of the first scientists who conceived a clear vision of Human Interaction with Machines. He expressed clearly his vision in two seminal papers: *"Man-Computer Symbiosis"*, IRE Transactions on Human Factors in Electronics, volume HFE-1, pages 4–11, March 1960 and *"The Computer as a Communication Device"*, Science and Technology, April 1968. In the latter article Licklider begins with a statement that may sound familiar today: "*In a few years, men will be able to communicate more effectively through a machine than face to face. That is a rather startling thing to say, but it is our conclusion*."

Here he meet the people working on the Project Mac. Probably during his first visit he started to think about the possibility of building a large nationwide network of computers committed to spread the new technological knowledge and to perfect the circulation of information.

Even if the project MAC was created to investigate a shared computing resource, a common utility designed to improve information management, unintentionally a second experiment was unfolded: an online storage system that became a way for people to share each other's data and programs. That secondary effect created not just a public utility as its creators were looking for but also an *information community* (Fano and Corbató, 1966). In fact, the system made possible for users to carry on a discourse with one another through the machine. The time-sharing computer system worked as a community pool of knowledge and skills on which anyone could participate according to his needs. As a matter of fact, one can imagine such a facility as an extraordinarily powerful library committed to serve an entire geographically distributed community. In short *an intellectual public utility*.

Two years before the publication of the Scientific American article "Time-sharing on Computers" in 1966[16], Project MAC had already embarked on a program to build such a system, called Multics[17], short for *MULTIplexed Information and Computing Service*. Reaching beyond MIT, the project also involved the Computer Department of the General Electric Company, which was building the computer's hardware, and the Bell Telephone Laboratories in New Jersey, which decided to collaborate with MIT following hopes that Multics would become the next central computing facility for the Labs. Meanwhile the work on Multics progressed at MIT. The first Multics became operational in October 1969 for general campus use and the management of its operation was transferred from Project MAC to the MIT Information Processing Center. By the end of 1971, the MIT Multics installation was operational twenty-four hours a day, seven days a week, and it served a community of more than 500 users. *Project MAC had truly created a public computer utility.* In 1968, Licklider replaced Fano as director and started a major thrust in human-computer interfaces and interactive modelling. In 1971, Ed Fredkin took over and started several new initiatives, including work in automated medical diagnosis. Michael Dertouzos became the fourth director of Project MAC in 1974. One of his first actions as director, in 1975, was changing the name of the project to LCS, the MIT's Laboratory for Computer Science. In 1975 the pioneering era of the Project Mac ended but it was obvious that its extraordinary legacy changed the future development of modern computing.

One of the most important results of the Multics system, or should I say of its technological

[16] Fano, R. M. and F. J. Corbato (1966). "Time-Sharing on Computers." Scientific American 215(3): 128-140

[17] *"Multics (Multiplexed Information and Computing Service) is a comprehensive, general-purpose programming system which is being developed as a research project. The initial Multics system will be implemented on the GE 645 computer. One of the overall design goals is to create a computing system which is capable of meeting almost all of the present and near-future requirements of a large computer utility. Such systems must run continuously and reliably 7 days a week, 24 hours a day in a way similar to telephone or power systems, and must be capable of meeting wide service demands: from multiple man-machine interaction to the sequential processing of absentee-user jobs; from the use of the system with dedicated languages and subsystems to the programming of the system itself; and from centralized bulk card, tape, and printer facilities to remotely located terminals. Such information processing and communication systems are believed to be essential for the future growth of computer use in business, in industry, in government and in scientific laboratories as well as stimulating applications which would be otherwise undone".* F. J. Corbató, V. A. Vyssotsky, Introduction and Overview of the Multics System (AFIPS1965)

breakthroughs, was the influence it had on the subsequent operating systems, the most prominent of which is undoubtedly UNIX. Moreover the legacy of the project Mac coupled with the legacy of the ARPAnet project fed more than one generation of computer scientists that produced in turn, the UNIX system, the BSD distribution 4.2, the TCP/IP protocol, the GNU project and lately the GNU/Linux operating system. This path is relevant for this history because it makes clear the strong link between the evolution of these technologies and the evolution of the object of this enquiry: the free/open source software.

Step three. The language: 1971 – 1982

It seems well worth going into some basic features of the operating system technology in order to properly understand what the words "open source" mean and why they have been used to define a broad system of software the foremost of which is Gnu/Linux. Therefore in the following paragraph I will briefly analyze the history of the UNIX[18] operating system that can be considered the technological ancestor of the present Gnu/Linux. More generally, I think it is necessary to analyse the idiosyncratic implications of developing a UNIX-like system, as Gnu/Linux is, in order to pinpoint its principal features in Design and Architecture and to understand its implication. In fact the use of a *style of writing software code*[19] certainly influences the system design as well as it affects its development. The style of Unix can be defined as "small-is-beautiful" philosophy, which is one of the basic technological determinants, or *rule of the language*, that influenced the development of the open source phenomenon. Although I do not want to deeply analysis the Unix operating system here, it seems useful to underline its basic and most important concepts.

The so-called Unix philosophy is based on the convincement that one doesn't need a complex interface in order to build up something complex but he can create any amount of complexity from the interaction of simple things. Following this idea Unix, or better its inventors, introduced the concept of "pipes": *channels of communication between minimal processes that are committed to build complex problem solving capability from simple logical units.* That principle is based on a *modular way* of problem solving which significantly characterizes the open source phenomenon. As a matter of fact GNU/Linux can be seen as a particular "clone" of the Unix system.

Ken Thompson, a young scientist at the Bell Lab, conceived the UNIX system in 1969. He

[18] The UNIX history is fascinating and inspiring for scholars studying Innovations in the software technology. Figure 4 is a useful map of UNIX development.

[19] The expression is based on a precise definition of what is the Software Source Code: *"Source code is any series of statements written in some Human-readable computer programming language. In modern programming languages, the source code, which constitutes a software program, is usually in several text files, but the same source code may be printed in a book or recorded on tape (usually without a filesyste). The term is typically used in the context of a particular piece of computer software. A computer program's source code is the collection of files that can be converted from human-readable form to an equivalent computer-executable form. The source code is either converted into executable by a compiler for a particular computer architecture, or executed from the human readable form with the aid of an interpreter".* Wikipedia definition of "Source Code" (http://encyclopedia.laborlawtalk.com/wiki/index.php/Source_code)

was deeply involved in the development of time-sharing systems when Bell Labs got away from the Multics research consortium leaving him with some Multics-inspired ideas about how to put up a file system, but without a proper collocation within a project. Meanwhile Dennis Ritchie and Doug McIlroy were working on a project involving Multics ideas and they didn't want to loose that capability too. So they decided to collaborate together in order to create a new operating system, using an informal method of development. As a consequence, UNIX was conceived by only three people and implemented by Ken Thompson alone.

The first important assignment of UNIX was the support of a kind of word processor for the Bell Lab patent department developed on a brand new PDP – 11[20]. The project sorted two results: the official one and an unofficial, probably more important, one, the development of an entire new operating system. The system gained a relevant success even if the creation wasn't in the Bell Labs and the AT&T plans and it was so well performing that 10 installations took place in 1971. The original Unix system was written in assembler and its applications in a mix of assembler and interpreted language called B, which was not able to fit, after the first success in simple tasks, its assignment. Therefore Dennis Ritchie added some critical features and started the development of a new language called C. Thompson and Richie succeeded in writhing a new operating system in only three years. Their 1974 paper on the Communications of the ACM gave Unix its first public coverage. In that paper the authors described the simple design of Unix and reported over 600 Unix installations. The system started to be famous.

At that point in time a critical non-technical event influenced the development of Unix and made possible the habit of sharing the software code of the Unix system. In fact, under a 1958 consent decree in settlement of an antitrust case, AT&T had been forbidden from entering the computer market. The Bell Lab was required to license UNIX at nominal price preventing AT&T from making it a "closed product". Therefore Ken Thompson was allowed to send tapes with the source code of Unix to anyone interested in it. Even if AT&T licensed Unix in several forms they couldn't build a real business on it and so it was the only system at that time that was distributed with the source code. The availability of the code turned to be the enhancing mechanism for the involvement of developers worldwide. Consequentially in 1978 the first Unix user group appeared. By that time Unix was used for operations support within the whole Bell information system and universities around the world also started to adopt and to study it[21]. The Unix operating system spread easily and rapidly.

[20] The PDP-11 was a 16-bit minicomputer produced by Digital Equipment Corp. in the 1970s and 1980s. The PDP-11 was used by several leading research programs in the 70s. It had uniquely innovative features and it was easy to program as a consequence of its modular design. As a matter of fact, the PDP-11 can be considered one of the most influential hardwares in the history of modern computing.

[21] It is worth making clear that point, because even though it is quite true that the Unix source code was easily available to researchers at universities, it is incorrect to say that its use for non-academic tasks has ever been "free-of-charge", even at the beginning of its history. Experts and researchers using Unix at the end of 70s and the beginning of 80s (as researchers at the Computer Science Department of the University of Turin) say that the AT&T provided the System with a licensing scheme, which allowed academic use but limited commercial or administrative use. Consequentially it is wrong to say that the system has always been "free" and it appears more appropriate to pinpoint the differences in the "fair use" permitted at that time, which was broader and more flexible than today. The real difference between Unix and other operating systems, typically the IBM's systems, was the distribution of the source code with the program. Consequentially it is not true that Unix was "free" but it is undoubted that scholars were "free" to tinker with its source code. It is also true, indeed, that the possibility of analyzing the source code created a strong and open community of peers interested in the development of Unix. Finally it is important to underline that

The same year the first Unix company, the Santa Cruz Operation, (SCO) was founded. Five years before another important player entered the development of Unix: the University of Berkleey. The campus had been famous as one of the most important spot of development since 1974. When Thompson had lessons at Berkleey campus he found an exciting environment of enthusiastic programmers able to appreciate the Unix revolutionary features. In particular there was a laboratory, led by the young student Bill Joy, which started the development of its own version of Unix called Berkleey Software Distribution (BSD). The first release of BSD[22] dates back to 1977. By this year there were two networks of developers operating on both Unix and BSD platform[23].

Then, in 1980 a new event influenced substantially the development of Unix, the Defence Advanced Research Project Agency (DARPA) was asking for a development team to implement its new protocol suite TCP/IP[24] under the Unix system. The choice was the BSD version of the University of California. The BSD version 4.2 released in 1983 was the first Unix version with TCP/IP's implementation and it testifies a substantial change in the networking capability of the entire system. Bill Joy, Andreas Bechtolsheim and Vinod Khosla, a Californian, a German and a Indian built in the same year, Sun Microsystems[25]. The pioneer company was formed with the strong commitment to produce a *dream Unix machine* with built-in network capability. To achieve the goal the founders decided to use the best hardware coming from the Stanford University combined with the BSD system developed by Bill Joy. The company that started as a "hacker-like" start up, tuned into a multinational corporation, which is today one of the top player in the market. By 1982 Unix had proved to be able to mix two powerful qualities: a stable and portable system plus an effective programming language but the rising status of Unix and its technical excellence created great commercial interest that threatened severely its open developing environment and modified its future development. In fact, the AT&T, released form legal constrains and knowing no other than the proprietary model for collecting profits from software, took control over its version of Unix. Contributions from universities disappeared threatened by lawsuits.

Moreover the new commercial players in the Unix market, seeking advantage by product differentiation, broke up its standard supporting system making difficult cross-platform compatibility and fragmenting the market. In addition the community of Unix developers, the

Unix reached remarkable success for the technical design more than for the licensing scheme.

[22] Berkeley Software Distribution (BSD) is the Unix derivative distribution created by the University of California at Berkeley in the 1970s. The name is also used collectively for the modern descendants of these distributions. Bill Joy, graduate student at Berkeley, released the first version in 1977. In June 1994 the 4.4 BSD split in two forms after a legal procurement committed to clarify the legal ownership of the system that was built upon the original AT&T Unix system. The 4.4 BSD-Lite was freely redistributable and contained no AT&T source while the 4.4BSD-Encumbered was available only to AT&T licensees. The closing release from Berkeley was 1995: the 4.4-Lite Release 2. Since then, several distributions based on 4.4BSD such FreeBSD have been maintained adopting the Open Source model. Furthermore it is relevant to point out that the tolerant nature of the BSD license has allowed many other operating systems, both free and proprietary, to incorporate BSD code, as the Apple Mac OSX.

[23] Figure 5 shows the map of UNIX and BSD systems development.

[24] A protocol suite is a set or group of communication protocols. The TCP/IP is the most important Internet suite of protocols. Its name comes from the two most important protocols in it: the Transmission Control Protocol (TCP) and the Internet Protocol (IP).

[25] The company's name SUN originally stood for Stanford University Network, which underlines the importance of network technologies in the corporate culture.

glorious hacker community of the 70s, was divided in two opposite sides, one called "longhairs" and a new one, coming from the homebrew computer club, called "shorthairs" hackers. The two wings disputed on technical issues while focusing on the exploding new Personal Computer market. While the shorthair faction was gaining momentum due to the PC success and its small proprietary software diffusion, the longhaired one was losing, probably more interested in technological issues than commercial opportunities. Following their natural technological attitude they misunderstood the PC revolution.

Despite these difficulties something happened in 1983 that created a second youth for Unix. A programmer/linguist, Larry Wall, invented the *patch utility*. The *patch* program enabled developers to cooperate by passing around incremental changes rather than entire code files. This was important because patches are much smaller than full files for patches apply even if the base file changes. Using that vital upgrade, flows of different project could coexist, evolve, differentiate and mix on a common code base. The *patch*[26] program did more than any other single tool to enable collaborative development over the Internet and its invention animated the whole Unix world. The importance of this invention suggested to Larry Wall to work on a new scripting language and in 1986 he began the development of Perl[27], which is now one of the leading languages of the open source system.

Today UNIX has become so complex and many different versions have appeared that it is more proper to define UNIX not as a single operating system but as a class of systems[28]. That is probably the clearest evidence of its success.

The analysis of the development of Unix provides important insights about the technological and sociological features of Open Source Phenomenon and it surely deserves more studies and researches. Following the chronological map I illustrated in this work, it started in the 60s and finished with the creations of several UNIX Companies at the beginning of the 80s, noticeably it finished when the Free Software movement began. Finally this story

[26] A short and effective definition of what patch is provided by Wikipedia: *"patch is a Unix computer program that applies textual difference between two programs and, more often, computer file or files containing such a difference, or "diff" files. The patch is typically an update meant to fix technical glitches like bugs or to improve the usability or the performance of the previous version of an application. Larry Wall, who also created the Perl programming language, wrote the original patch. patch is capable of applying various formats of diffs and has become quite popular among developers who frequently exchange a small change in programs with fellow programmers. patch has become especially popular within the Free Software community as it allows uninitiated developers to contribute to a project. Most open source projects are hosted on a public "versioning service system" from which everyone can download the source code. But for obvious security reasons, only the developers of the project can be trusted with write access. patch makes it possible for developers that do not have write access to develop the code on their own, make a patch out of it and send it to the main developers."* (http://en.wikipedia.org/wiki/Patch_%28Unix%29)

[27] As defined in the PERLINTRO(1) Perl Programmers Reference Guide: *"Perl is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI develop-development, and more. - The language is intended to be practical (easy to use, efficient, com-complete) rather than beautiful (tiny, elegant, minimal). Its major features are that it's easy to use, supports both procedural and object-oriented (OO) programming, has powerful built-in support for text pro-processing, and has one of the world's most impressive collections of third-party modules."*

[28] In fact a Single UNIX Specification (SUS) exists, which describes the family of standards for operating systems to qualify for the name "Unix". The SUS specifies: *" The user and software interfaces to the OS are specified in four main sections: 1- Base Definitions - a list of definitions and conventions used in the specifications and a list of C header files which must be provided by compliant systems. 2- Shell and Utilities - a list of utilities and a description of the shell, sh. 3- System Interfaces - a list of available C system calls which must be provided. 4- Rationale - The explanation behind the standard. The standard user command line and scripting interface is the Bourne Shell. Other user-level programs, services and utilities include awk, echo, and numerous (hundreds) others. Required program-level services include basic I/O (file, computer terminal and network) services. A test suite accompanies the standard. It is called PCTS or the Posix Certification Test Suite. Note that a system need not include source code derived in any way from AT&T Unix to meet the specification."* See Wikipedia: http://en.wikipedia.org/wiki/Single_UNIX_Specification#Specification

clarifies the relationship between the Unix-like open and modular system of development and the open source phenomenon: 1) it proves how important the design and architectures have been as well as the software language in the development of the phenomenon, 2) it shows the strong success of an open community in managing software development, 3) it illustrates the genealogy of the typical open source way of thinking about computer programming.

Step four. The birth of the PC: 1977 – 1991

The years from 1977 to 1991 are the turning point of the history I am analysing here because a number of innovations that deeply influenced the evolution of modern computing appeared during this period. In only seven years, from 1977 to 1984, several new technologies were developed, new corporations succeeded in new markets and new legal institutions appeared.

I choose to set the starting point of this period on the 3rd of January 1977 when Steve Jobs, Steve Wozniak and Mike Markkula founded the Apple Computer, because of the importance that this corporation had not only for the personal computer market but also for the computer industry as a whole. The period comprehends, among others, the commercialization of the first Apple Macintosh in 1984, the outstanding achievements of the Microsoft Corporation, the development of the IBM "project chess" led by Philip Don Estridge, which created the first *IBM-PC*[29] as well as the success of the clones-PC, based on x86 Intel's hardware architecture. This phase finishes in 1991 with the creation of the World Wide Web by Tim Barners-Lee that unveils an entirely new technological scenario based on a new powerful distributed communication network.

Therefore the following pages focus on the diffusion of the so-called Personal Computer. Even if it is well known that the years from the end of the 70s to the end of the 80s have been extremely important for the ICT industry, I will try to point out how and why the technological economic and legal events occurred during these years influenced the open source phenomenon. In fact the period set a central change in the history of computers, which I can roughly call "the proprietary change" and free software came as a reaction to that. Two main reasons contributed to the personal computer uprising: the definition of a standard architecture in the hardware industry by the *IBM-PC* and a similar position acquired by Microsoft in the software market. As a matter of fact the GNU project and the Open Source Initiative were built as the alternative to the Windows-*IBM PC*.

It is interesting to notice how fast, during this period, the personal computer industry passes through different phases and set on a "Specific Phase" (Utterback, 1994). By the same token, it is easy to see that major innovations are concentrated at the beginning of the period while they are likely to disappear at the end. As Utterback points out, this dynamic of change is natural for many "assembled products", as the Windows-PC is, but in this case it has been remarkably fast. It is also important to take into account the dominant design emerged during this "specific

[29] It is worth noticing that the IBM PC™ (*Personal Computer*) is a trademark of International Business Machines, so using this name I intend to identify a specific architecture and design.

phase" of the industry as it reveals where the products differentiation of the GNU/Linux system comes from. As a matter of fact Microsoft acted as an excellent "mechanistic" firm but they faced the same problems this kind of organizations usually have. With a high degree of specialization of labour and a market driven innovation process, they missed creative capability. As a matter of fact the diffusion of the Microsoft-PC forced Microsoft to perfect marketing related activity, like technological support or advertisement. Moreover they concentrate on enforcing IPR protection of software programs in order to prevent the appropriation of their design, more than sharing and building an open community. They actually focused on the best way to produce and sell software products and not necessarily on the best way to develop them.

Although the unambiguous birth of the modern PC is usually set in 1984 with the release of the first *IBM-PC*, the technical ideas the personal computer is based on, as I showed earlier in this chapter, date back at least to the 50s[30]. Moreover the term "personal computer" was in use since the beginning of 70s in the famous Xerox Researches Lab in Palo Alto California. Therefore it may be useful to briefly explain the generation of that idea. As Swann (1997) explains before 1951 it is almost impossible to find out marks of the modern PC. In that year Remington Rand produced the first commercial computer (UNIVAC 1). The computer industry emerged from the tabulating equipment industry that was dominated by IBM. During the 60s the picture doesn't change and IBM hits a share market of 75% in computer installations but two demanding challenges appeared as the clear effect of an ongoing technological revolution. The first was the continuing step decline in the cost of achieving a given level of data processing performance and the second was the so-called "plug-compatible" computer. These two trends were the main concerns for IBM. The alarm became certitude in 1971 when the Intel Corporation invented the microprocessor. The "chip" was the key innovation that revolutionized the entire architecture of computer machines and permitted to expand exponentially the power of data processing. This technological improvement plus the decreasing costs of components production began to suggest a new design of computer machine: the "small and cheap" personal computer.

In 1974 Intel introduced the 8080 microprocessor[31] and in 1975 a small company based in New Mexico produced the famous MITS/Altair[32]. Ed Roberts hobbyist who was trained in

[30] As Langlois (1992) points out the history of computers is much more longer: *"Although the history of computers dates back at least to the mechanical tinkering of Charles Babbage in the nineteenth century the electronic digital computer was the product of the Second World War. In Novermber 1945, J. Presper Ecjert and Jhon W. Mauchly of the Moore School at the University of Pennsylvania produced the ENIAC (electronic numeral integrator and computer) the first all-electronic digital computer under contract with the U.S. Army. The machine took up 1,800 square feet, boasted 18,000 tubes, and consumed 174 kilowatts. Collaboration with the mathematician John von Neumann led a few years later the idea of a stored-program that is a programmable rather than a special purpose computer, an approach called the Von Neumann architecture that is used almost universally today in computers of all sizes."*

[31] *"The Intel 8080 was an early microprocessor designed and manufactured by Intel. The 8-bit CPU was released in April 1974 running at 2MHz, and is generally considered to be the first truly usable microprocessor CPU design".* See Wikipedia, http://en.wikipedia.org/wiki/Intel_8080

[32] *The MITS Altair 8800 is a microcomputer design from 1975, based on the Intel 8080A CPU. Sold as a kit through "Popular Electronics" magazine, the designers intended to sell only a few hundred to hobbyists, and were surprised when they sold over ten times that many in the first month. Today the Altair is widely recognized as the spark that led to the personal computer revolution of the next few years: The computer bus designed for the Altair was to become a de facto standard in form of the S-100 bus, and the first programming language for the machine was Microsoft's founding product, Altair BASIC.* See Wikipedia, http://en.wikipedia.org/wiki/Altair_8800

model airplanes control devices created the machine, another proof of the importance of the user innovation model in the development of personal computer.

It is in 1977 that one can start to use the word PC without any misunderstanding. In fact in early 1977 the standard operating system of Intel 8080 architecture was ready: the CP/M[33]. The creation of this system allowed the development of the first software for PC dragged by an enthusiastic community of hobbyists, that I called in the previous paragraph shorthaired hackers, that were keen to share their knowledge regarding the new exciting technology; two of the most industrious were William Gates and Paul Allen. However the CP/M did not became a standard because in 1977 a new machine was introduced influencing not only the technical competition within computer manufactures but primarily and most importantly the general idea of personal computer: the Apple II. The machine, designed by Steve Wozniak and Steve Jobs is still one of the most celebrated example of innovation in the whole history of personal computer. Before founding the Apple Computer Corporation, in early 1977, both Steve and Steve were employed in different company that refused to follow their technical inspiration, the former was an engineer at HP and Atari employed the letter. The big new idea of Steve & Steve was to build a machine with an open-ended architecture. But it is now clear that the remarkable success of the company was achieved because of its marketing intuition about what the next ready-to-go personal computer was expected to be. More precisely the Graphical user interface, introduced with the Macintosh and cantered on the concept of "what you see is what you get"[34] design, was one of the most important innovations of Apple.

The "Project Chess". Until 1981 Apple, Commodore and Tandy were the three top makers of microcomputer while other incumbents seemed to be left behind. The most powerful one was IBM. As a result in august 1981, threatened by the new technological landscape, IBM succeeded in producing the IBM-PC that set the paradigm thereafter. The project was secret and its name was "project chess". Philip Don Estridge was appointed as chief engineer with a clear commitment: creating a personal computer that could be easily used by the masses. He led the team, which developed the IBM PC and he is probably responsible for choosing to have an

[33] CP/M (translated variously as Control Program/Monitor, Control Program for Microcomputers or Command Processor for Microcomputers) is an operating system for Intel 8080/85 microcomputer that was created by Digital Research, Inc. founded by Gary Kildall. The mix of CP/M and S-100 bus computers installed on the MITS Altair was the first "industry standard" for personal computer and it was widely used through the late 1970s and into the mid-80s.

[34] WYSIWYG is an acronyms for What You See Is What You Get. It is used in computing to describe a seamlessness between the appearance of edited content and final product. So, it defines a characteristic design. *"The phrase was originated by Jonathan Seybold and popularized at Xerox PARC during the late 70s when the first WYSIWYG editor, Bravo was created on the Alto. The Alto monitor (72 pixels per inch) was designed so that one full page of text could be seen and then printed on the first laser printer. When the text was laid out on the screen 72 PPI font metric files were used, but when printed 300 PPI files were used — thus one would occasionally find characters and words slightly off, a problem that continues to this day. (72 PPI came from the standard of 72 "points" per inch used in the commercial printing industry.) Seybold and the researchers at PARC were simply re-appropriating a popular catch phrase of the time originated by "Geraldine", a character on The Flip Wilson Show (1970-1974). In addition to "What you see is what you get!" This character also popularized "The Devil made me do it!" The Apple Macintosh system was originally designed so that the screen resolution and the resolution of the dot-matrix printers sold by Apple were easily scaled: 72 PPI for the screen and 144 DPI for the printers. Thus, the on-screen output of programs such as MacWrite and McPaint and were easily translated to the printer output and allowed WYSIWYG. With the introduction of laser printers, resolutions deviated from even multiples of the screen resolution, making WYSIWYG harder to achieve. Charles Simonyi, the PARC researcher responsible for Bravo, joined Microsoft in 1981 to start development of application programs at Microsoft. Hence, Bravo can be seen as the direct ancestor of Microsoft Word."* See Wikipedia: http://en.wikipedia.org/wiki/WYSIWYG

open architecture and buying parts and the software outside IBM. His choice of an open architecture and off-the-shelf components resulted everywhere in the IBM PC architecture. One of the main consequences was the choice of the PC-DOS operating system, produced by a new Seattle-based company: The Microsoft Corporation. Another central result of his liberal approach was that IBM allowed Microsoft to licence MS-DOS to other computer manufacturers. In fact, even if this hazardous decision did not prevented IBM competitors from selling their MS-DOS based computers, it helped the establishment of the IBM standard and actually it sanctioned the huge success of Microsoft Corporation. Furthermore the decision led to the creation of the so-called IBM "clones". It may be interesting to point out that the first success of the clones was due to excess demand of personal computers that IBM couldn't satisfy rather than IBM's weak control over the operating system[35]. The clones appeared at the beginning of the 80's but even if they didn't have great diffusion until the end of the decade however a new powerful and long lasting industry was born in 1981.

Apple and Lotus: GUI and spreadsheet. The years from 1981 to 1984 viewed an explosion of new products but in 1984 two relevant facts happened that are meaningful in our history: the Lotus software and the Apple Macintosh came into view.

Lotus. Partners Mitch Kapor[36] and Jonathan Sachs set up Lotus software in 1982. Lotus' first product was a software for electronic presentation of the Apple II platform, known as Lotus Executive Briefing System, but the company is more broadly known for its groundbreaking Lotus 1-2-3 spreadsheet applications released in January 1983. Moreover the importance of Lotus is also related to the introduction of the first multimedia software such as Ray Ozzie's Symphony and the Jazz office suite for the Apple Macintosh computer. That combination of Mac hardware and Lotus software created the first "multimedia suit" for Personal Computers. Moreover Lotus acquired many software companies in order to extend its products base such as Freelance Graphics, Ami Pro, Approach, and Organizer. In the late 80s, Lotus developed Lotus Magellan a file management and indexing utility. In the early 90s, several of the products were packaged together under the name of "Lotus SmartSuite" but although it was initially more popular than Microsoft Office, Lotus lost its dominance in the desktop applications market. In 1990 as consequence of his hacker beliefs Mitch Kapor co-founded the "Electronic Frontier Foundation"[37], which is a non-profit organization working in the public interest to protect privacy, free expression, and access to public resources and information online, as well as to promote responsibility in new media. Once more the evolution of computer history is

[35]Langlois (1992), Externals Economies and Economic Progress: The Case of the Microcomputer Industry, The business History Review, Vol. 66, No. 1, High-Technology Industries, 1-50.

[36] The funder of Lotus, Mitch Kapor is what we can call a "hacker" with great technological skills and clear ideas about the importance of freedom in digital communication. In particular he was afraid of the personal information control produced by the new digital machines. Because of that his main research field has been the encryption of codes. The matter is particularly relevant if one would consider the privacy-related problems concerning the pervasive use of networked digital system (Google being the foremost example).

[37] The EFF is a no-profit organization committed to defend "digital rights" and to protect an open technological environment for Innovation. As reported on the EFF's web site: *"The Electronic Frontier Foundation (EFF) was created to defend our rights to think, speak, and share our ideas, thoughts, and needs using new technologies, such as the Internet and the World Wide Web. EFF is the first to identify threats to our basic rights online and to advocate on behalf of free expression in the digital age."* Electronic Frontier Foundation (http://www.eff.org/about/)

related to a hacker which turned out to be also a very good businessman but which has always cared about the defence of an open technological system.

Macintosh. The creation of the Macintosh GUI has probably been one of the most influential elements setting the standard of human interaction with machines in consumer PC architecture until now. Jef Raskin is probably the leading character in the creation of Apple Macintosh. Raskin met Apple Computer's Steve Jobs and Steve Wozniak for the first time at a public presentation of their Apple II personal computer in 1977 and he found the right partners to start the Macintosh project. This computer was comparable in power to the Apple II with a small 9-inch black-and-white display built into a small case with a floppy disk. The main "new idea" was the creation of the best user interface ever produced in companion with a new powerful, small and "plug and play" machine. As a result of this clear commitment they designed the Mac's Interface, which is likely to be the most innovative achievement of Apple Computer and still the main competitive advantage of Macintosh Computers – it is interesting to notice that after 2000 the Macintosh system merged with the UNIX operating system. The core design came from the researche projects on GUI (Graphic Users Interface)[38] undertaken by the Xerox Lab (in particular on the Alto System)[39] and once again, in that history, the connection between the community of researchers and the production of innovation is confirmed. Raskin left Apple pretty soon but his influence persisted thereafter. He has been one of the most influential personalities in user interface researches field as the text *"The Humane Interface"* clearly shows, where his ideas about human-computer interaction are well expressed. As a matter of fact his conception of Mac GUI still survive on the desktop of all Mac Users worldwide.

The Apple Macintosh was launched in 1984 with a now famous advertisement based on the George Orwell's book "1984": *"On January 24, Apple Computer will introduce Macintosh. And you'll see why 1984 won't be like '1984'."* The evident implication was that the Mac's new, "user friendly" GUI would set computing free from an elite of large corporations and technocrats (the foremost example is the IBM but also DEC). In spite of such a messianic announcement, this didn't happen. Microsoft got several Macintosh prototypes in 1983, before the Macintosh official launch and they went meticulously into its most interesting features. In 1985 Microsoft launched Microsoft Windows with its own GUI for IBM PCs copying many elements of the Macintosh OS. This created a long-lasting legal battle between Apple Computers and

[38] The GUI (graphical user interface) is a method of communication and interaction with computer through a metaphor of direct manipulation of graphical images and things in addition to text. Douglas Engelbart at the Stanford Research Institute invented the precursor of present GUIs with the "On-Line System" project. It was built upon the use of text-based hyperlinks manipulated with a device called "mouse". The concept of hyperlinks was further refined and extended to graphics by researchers at Xerox PARC who went beyond text-based hyperlinks and used GUIs as the primary interface for the Xerox Alto computer. Most modern general-purpose GUIs are derived from this system. Examples of systems that support GUIs are Mac OS, Windows or the FLOSS GUI Gnome.

[39] I think it interesting to report a part of an interview with Jef Raskin where he finally explains the difference between the system developed at the Xerox PARC and the Mac Operating System: *"BYTE: What's the other side? In what sense is Macintosh a departure from what was been done and thought of at PARC? RASKIN: Theirs was all based on Smalltalk and had a different model of what the user interface would look like. I thought they had a lot of good ideas. The difference between Apple and PARC is that Apple was designing things to be sold in large quantities and PARC designs things to play with. While they weren't concerned with questions of production, I very much was"* Ref: http://rchi.raskincenter.org/aboutrchi/aboutjef.php published on Byte issue 8/1984, pp. 347-356.

Microsoft, ending with an out of court settlement. In this decision it was stated that Microsoft would be granted access to and allowed unlimited use of the Macintosh GUI. Although the first version of Windows was probably technically poorer than the Mac, a Windows equipped PC clone was normally cheaper. Moreover there has always been more software available for Windows because of the open nature of the PC platform. These two elements succeeded in the new market and Microsoft set the standard for personal computers until a new turning point occurred in 1991: the creation of "the World Wide Web".

The World Wide Web. The origins of the Web came from the CERN laboratories where, at the end of the 80s, Tim Berners-Lee and Robert Cailliau worked on the ENQUIRE project, acronyms of Enquire Within Upon Everything. While it was rather unlike the Web as we know it today, it enclosed several core ideas of the contemporary WEB. Berners-Lee says that much of the inspiration behind the project comes from the attempt to access libraries of information that were spotted on several different servers at CERN. He published the proposal for the World Wide Web on November 12, 1990 and wrote the first web page the day after on a NeXT workstation. By December 1990, Berners-Lee built the necessary instruments for a working Web: the first web browser (which was a web-editor as well) and the first web server. On August 6, 1991, he posted a summary of the WWW project on the alt.hypertext newsgroup: it was the first public announcement.

The Web is based on the organizational model of the "hypertext", which come from an outstanding legacy as we have seen before, but Berners-Lee's brilliant advance was to connect the concept of hypertext to the "distributed network architecture" developed by the ARPAnet project since the 60s. As a consequence he developed a system of globally unique identifiers for resources on that network, namely the Web, and elsewhere: the Uniform Resource Identifier. The system was different from other hypertext systems produced before, in particular, it required only unidirectional links rather than bidirectional ones. Moreover, unlike other previous programs, the World Wide Web was non-proprietary, making it possible to develop servers and clients independently and to add extensions without licensing restrictions. On April 30, 1993, the CERN announced that the World Wide Web would be free for anyone.

As one can easily see the creation of the WWW was a new, important step along the way of increasing communication capability between connected computers and that it is likely to have been the natural evolution of the technological paradigm started several decades before. It was conceived as an open architecture base on open standard, freely available to anyone. As a matter of fact the Web is made up of three standards: 1) The Uniform Resource Locator (URL)[40], which specifies how to address each page of information and where it can be found; 2) Hyper Text Transfer Protocol (HTTP)[41] which specifies how the browser and server send the

[40] As defined in the Internet Standard (RFC 1738) written by T. Berners-Lee, L. Masinter and M. McCahill in 1994 *"Just as there are many different methods of access to resources, there are several schemes for describing the location of such resources. The generic syntax for URLs provides a framework for new schemes to be established using protocols other than those defined in this document. URLs are used to `locate' resources, by providing an abstract identification of the resource location. Having located a resource, a system may perform a variety of operations on the resource, as might be characterized by such words as `access', `update', `replace', `find attributes'. In general, only the `access' method needs to be specified for any URL scheme."*

[41] As is defined in the Internet Standard (RFC 2616) written by R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L.

information to each other; 3) Hyper Text Mark-up Language (HTML)[42], a method of coding information in order to display them on a variety of devices.

This standard architecture made participation and development easier between programmers geographically distributed and supported a vast community of users. The web increased exponentially and the idea of public knowledge utility proliferated worldwide. Furthermore following an open-science philosophy, the World Wide Web Consortium (W3C) was founded as the legal institution committed to manage the development of these standards. Thirty years after the Project MAC, the effort finally achieved an outstanding result.

Step five. The GNU Project and the Open Source Initiative: 1983 – 1998

GNU is neither Unix nor a "free beer"

The history of free-software is undoubtedly related to the personality of Richard Matthew Stallman, a renowned hacker who, during the first half of the 80s, bravely reacted to the commercialisation of UNIX creating a new Operating System, in which key power was the free access to the source code, *the freedom of tinkering*. The Stallman's project answered the request of a large part of the scientific community related to the Academia that was used to thinking of software development as a commons-based activity committed to develop a Public Utility of Computer machines. But that approach, as we saw in the previous paragraph, was severely threatened by the advent of the proprietary software market. However, even if several computer scientists worldwide perceived the threat of closed systems, only Richard Stallman was so confident, brilliant and good at programming to try to answer. In fact, Stallman was shocked by the idea that the next generation of software could have been closed by proprietary instances stopping free communication and knowledge sharing between developers. The proprietary systems, in fact, weren't distributed with the source code and that means nobody could access the way they worked. Reacting in a hacker fashion, he started the challenging development of an entire new Operating System, called GNU (Gnu's not Unix) as primary technical instrument to preserve the open development model - the natural expression of the hacker tradition.

Richard Stallman founded the GNU project in 1984 and The Free Software Foundation[43] in

Masinter, P. Leach, T. Berners-Lee in June 1999 about "HTTP/1.1": *"The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol, which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers [47]. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred. "*

[42] HTML is a standard under the ISO/IEC 15445:2000(E) standard documentation edited in may 2000. One of the first definitions of HTML can be found on the RFC 1866 - The hypertext markup language 2.0: *"The Hypertext Markup Language (HTML) is a simple markup language used to create hypertext documents that are platform independent. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of domains. HTML markup can represent hypertext news, mail, documentation, and hypermedia; menus of options; database query results; simple structured documents with in-lined graphics; and hypertext views of existing bodies of information."*

[43] The Free Software Foundation (fsf) is a no-profit organization committed to promote free software. As is said on the fsf website: *" Free Software is a matter of liberty not price. You should think of "Free" as in "Free Speech". The Free Software Foundation (FSF), established in 1985, is dedicated to promoting computer users' rights to use, study, copy, modify, and redistribute computer programs. The FSF promotes the development and use of Free Software, particularly the GNU operating*

1985. In the same year he also created, with Eben Moglen, a Law professor at the Columbia University, the General Public Licence as the necessary legal instrument to protect the projects. This foundation can be interpreted as the expression of the need to *institutionalize the Hacker philosophy*, which used to be the common attitude of developers but that was fatally threatened by the growing diffusion of proprietary software. The interoperability among different computer machines, chiefly induced by the *IBM-PC* standard, endorsed the creation of software that can easily run on different machines, manufactured by competing corporations. As a consequence a new market for closed software was established. In the same way, it was clear that the appropriability of the technological knowledge embedded in the software program, were essential in order to sell it as a "product" and consequentially to make profits on it. This new technical possibility generated a new stream of product innovation; it definitely was a central turning point for the software industry and the creation of FSF was the answer of an eccentric and skilful part of the software developers' community. However what is special about Mr Richard M. Stallman is that he understood before any other that the proliferation of proprietary software was the sign of a radical change in the way of thinking about software in general. He realized that something was changing the practice of sharing technical information about software programs and he felt the need of doing something about it. Thus he started to operate in order to preserve the tradition of "sharing the code" that characterized the world he used to know and to like.

It is worthwhile briefly considering Stallman's professional training which took place at the MIT Artificial Intelligence (AI) Lab during the 70s, in order to better understand the reasons why Richard developed its intense hacker consciousness. He joined the MIT AI Lab in 1971 and therefore he grew up in an open environment where the practice of sharing software code was normal. Because the software was typically embedded in huge and expensive computers, nobody thought that it could be sold as a stand alone product and it was not rare that the source code was available to liberally discuss its possible improvement. Therefore the idea that exactly the same system could run on several different hardware platforms was uncommon and the practice of freely revealing the software source code was not only harmless but also actually useful. In fact the possibility of sharing technical information enhanced a "distributed revision process" that induced the production of a massive amounts of information about the development, so this practice was a central feature to improve the possibility and the effectiveness of the software innovation process. Because of that after the proprietary turn of UNIX, Richard realised that the intellectual habit of sharing the software code needs to be protected but in order to accomplish the task he basically needed a proper instrument. The tools were 1) *the GNU project* and 2) *the GNU General Public Licence,* which express the concept of "copyleft". The former was the technological basis of the new and free operating system; the latter was the legal shell to keep it free, a new form of licence, which implied the obligation to share the software code. The sum of these two factors led to the birth of the free software

system, used widely in its GNU/Linux variant. The FSF also helps to spread awareness of the ethical and political issues surrounding freedom in the use of software." (http://www.fsf.org/)

phenomenon, as we know it today. In fact Stallman understood that a free operating system was needed with a new licensing scheme able to protect its freedom. His chief point was that closing the software was a mistake because of technical and ethical motivations: the main technical reason is that the unwillingness to share the code limits its technical communication and consequentially reduces innovation, the ethical one claims that the software must be freely available in order to spread both technological progress, by connecting as much knowledge resources as possible, and social progress, by making the access to information easier for everyone. In this view the software is an extension of our natural language and as such calls for a free path of growth.

The GNU project, which is the technological base of the GNU/Linux operating system, started with a public announcement written by Richard Stallman to the MIT Lab newsgroup, Tue, 27-Sep-83 at 12:35:59. The title of the message was "new UNIX implementation"[44]. Richard had a clear idea of the project and a detailed plan to manage it. The relevance of this announcement is much important when we notice that the substance of the free software movement has not changed thereafter. The general idea was to create a new operating system, which was free and based on UNIX, but not UNIX[45]: something new, reliable and totally free. Therefore from the beginning the project meant to build a new system based on Unix with some improvements, and to distribute it for free. However the notion of free, despite its simplicity, is not obvious and in particular the word "free" could be misunderstood. Consequentially the idea of free-software was extremely well codified by Richard. The main problem was to create a new licence, enforceable and reliable, in order to keep the software free. In fact the GNU project needed to be protected from proprietary appropriation but Stallman did not trust the usual set of licensing terms available. The goal was to make the software free but not to let it loose in Public Domain. So he developed the General Public Licence committed to preserve four degrees of freedom: "*The freedom to run the program, for any purpose (freedom 0). The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this. The freedom to redistribute copies so you can help your neighbour (freedom 2). The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.*"[46] The description of these four freedoms is the main feature of free-software. It represents

[44] *"Free Unix! Starting this Thanksgiving I am going to write a complete Unix-compatible software system called GNU (for Gnu's Not Unix), and give it away free to everyone who can use it. Contributions of time, money, programs and equipment are greatly needed"*. See: http://groups-beta.google.com/group/net.unix-wizards/msg/4dadd63a976019d7?htlm=en

[45] As he said in the same message: *"GNU will be able to run Unix programs, but will not be identical to Unix. We will make all improvements that are convenient, based on our experience with other operating systems. In particular, we plan to have longer filenames, file version numbers, a crashproof file system, filename completion perhaps, terminal-independent display support, and eventually a Lisp-based window system through which several Lisp programs and ordinary Unix programs can share a screen. Both C and Lisp will be available as system programming languages. We will have network software based on MIT's chaosnet protocol, far superior to UUCP. We may also have something compatible with UUCP."* See: http://groups-beta.google.com/group/net.unix-wizards/msg/4dadd63a976019d7?htlm=en

[46] As the GNU web site reports: "*We maintain this free software definition to show clearly what must be true about a particular software program for it to be considered free software. ``Free software'' is a matter of liberty, not price. To understand the concept, you should think of ``free'' as in ``free speech,'' not as in ``free beer.' Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software: […] A program is free software if users have all of these freedoms. Thus, you should be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere. Being free to do these things means (among other things) that you do not have to ask or pay for permission*." See:

the foremost institution, which ruled and still rules the movement. Moreover these four-degrees of freedom are the chief principles the free software system of licences is built on and the defence of these freedoms can be seen as the central commitment of the Free Software Foundation. With this accurate definition of "openness" in software, Richard Stallman originated the Free Software movement, which lately generated the Open Source Definition and influenced all the open model of software development thereafter.

Eric Raymond and the Open Source Initiative

By the definition of Open Source Phenomenon I aim to group together three different entities: a) the GNU project with the Free Software Foundation, b) the Open Source Initiative and c) the Operating System GNU/Linux. All of them are important to understand this case study and so it is natural to continue this historical investigation switching from the free software to the open source software. The following section is committed to describe the foundation of the Open Source Initiative.

The "Open Source" label came out for the first time during a meeting held on February the 3rd 1998 in Palo Alto, California. The persons involved were Todd Anderson, Chris Peterson, John Hall, Larry Augustin, Sam Ockman and Eric Raymond. The Netscape Corporation that was reacting to the diffusion of Microsoft Internet Explorer hired the team in order to get valuable suggestions on the best way to open the source code of its Netscape Browser. In fact Raymond and his colleagues had a new vision about the future of the open development model and the Netscape announcement created an important occasion to test the superiority of the open development process. More important, Raymond and the others realized that it was time to leave the *antagonistic attitude* that characterized the "free software" movement in the past. Following this convincement new tactics and a new label were created. The "Open Source Software," contributed by Chris Peterson, was the definition of this new attitude. After February 1998, the open sources team worked on diffusing the term among developers worldwide. Linus Torvalds gave an all-important imprimatur to the idea, Bruce Parens offered to trademark the "open source" and to host the web site; Phil Hughes offered a stage on the Linux Journal and Richard Stallman flirted with the term even if he changed his mind later.

The key features of the Open Source movement are defined by the Open Source Definition, which derives from the Debian Free Software Guidelines[47]. Bruce Parens composed the original draft; it was refined using suggestions from the Debian GNU/Linux Distribution[48] developers' community during an e-mail conference in June, 1997. They then voted to approve it as Debian's publicly stated policy. It was revised in order to remove the Debian-specific references

http://www.gnu.org/philosophy/free-sw.html

[47] See: Debian Social Contract, Version 1.0 written on July 5, 1997. (http://www.debian.org/social_contract#guidelines)

[48] See http://www.debian.org/ : *"Debian is a free operating system (OS) for your computer. An operating system is the set of basic programs and utilities that make your computer run. Debian uses the Linux kernel (the core of an operating system), but most of the basic OS tools come from the GNU project; hence the name GNU/Linux. Debian GNU/Linux provides more than a pure OS: it comes with more than 8710 precompiled software bundled up in a nice format for easy installation on your machine."*

and it was accepted as official Open Source Initiative in February 1998. Established in 1998, the Open Source Initiative is now a California public benefit (not-for-profit) corporation, which is held by board members who lead its directorship.

The story of the creation of the Open Source movement takes place between 1998-1999. On the 22nd of January 1998, Netscape announces it will release the source code for Navigator and create a task force to manage the project. The following February in Palo Alto a crucial brainstorming session took place and the choice of the term "Open Source" was made. During the following week, Bruce Perens and Eric Steven Raymond launched www.opensource.org. Since the fist arrival of the term there weas an animated debate within the hacker community: "open source" vs. "free software". This terminological argument was understood by all parties to be a proxy for a wider issue about the community's relationship to the business world. In the same months an important Netscape press released referred to "open source" and at the same time O'Reilly Associates, a famous editor of technical books on computer science, agreed to use the term on their web page. The same month references to "open source" appeared in the trade press. The 22nd of Jun 1998 IBM declared that it would sell and support Apache, a renowned open source web server, as part of its WebSphere suite. In July The Economist magazine took editorial notice of Linux, reporting positive findings on the performance of the open source model. Moreover Oracle announced that they would port their databases to Linux and Forbes magazine published a major article on open source, with Linus Torsvald on the cover. For the first time the combination of the Linux operating system and the opens source methods were seen as a real model to perform business in the software industry. Meanwhile Sun Microsystems made Solaris available under a free license to individual users, also to educational/non-profit/research institutions. The uprising of Open Source was clear.

As a consequence of the growing importance of the open source movement, on the 11th of Aug 1998 the Revision 1.0 of the VinodV memorandum on open source circulated inside Microsoft. The secret document, better known as the Hollowing Document, produced by Microsoft officers suggested how to contrast the phenomenon in legal and even non-legal ways. In August SCO too joined Linux International. On September 1998 Red Hat announced that Intel and Netscape had acquired a minority stake in the leading Linux distribution and the new system reached such popularity that Microsoft issued a statement adducing Linux's existence as evidence that Microsoft did not in fact have the monopoly of the operating system market. As proof of Microsoft's particular interest in open source on the 1st of November 1998 the Halloween Document, documenting Microsoft's plans to stop Linux and other open-source projects, became unintentionally public. A weeklong furore exploded in the US media because of the tactics described in the document. In December 1998 IDG announced that Linux market share had increased 212% in 1998. 1999 began with another success of open source because the 27th of January the HP and SGI announced, on the same day, that they would support Linux on their machines, following a trend unveiled earlier by Sun. In February 1999: IBM announced Linux support of its hardware, a Lotus port for Linux, and a partnership with Red Hat. The

first LinuxWorld, which was the first Linux and Open Source real trade show, took place at the beginning of March 1999 and major announcements by HP, IBM, SAP signalled the beginning of serious corporate support. Moreover on the 15th March 1999 Apple released Darwin, which can be viewed as a new substantial opening for Open Source in the Mac OSX operating system. By that time Microsoft claimed Linux was outselling Windows 98 at major software retail outlets. By the end of 1999 the Open Source phenomenon conquered the status of a reliable operating system and its diffusion boosted as never before.

The main goal of the open source movement has been to fix the univocal meaning of what is open source by creating the Open Source Definition. This definition was designed in order to capture the great majority of the software that was originally meant to be open source and it intended to define some precise criteria to identify which licence was open and which wasn't. However, since the term became widely used, its meaning lost some precision and consequentially the Open Source Initiative invented the OSI Certified mark as the way of certifying that the license under which software is distributed conformed to the Open Source Definition and could be called open source. The Open Source Definition version 1.9 outlines the requirement a licence must fit to be considered an Open Source License, in particular the definition is built upon 10 criteria[49]. The Open Source Definition set the characteristic of the Open Source community and it also fixed a new way of thinking about the open development model (Raymond 1997). It was the commencement of a new idea of openness that could suit a broader audience than free software because it was more flexible. In fact after 1998 the word Open Source has become the common definition of the movement, which started to comprehend not only the software community but a wide group of fellows from different disciplines and, even more important, many big software corporations.

Step six. Linux age: 1991 - 2001

The last building block of the history I address here is the birth of Linux. Its story is centred on the character of Linus Torsvald and it shows how his intuition, empowered by a broad community, has built one of the most important software programs ever designed.

Linus Torsvald was a computer science student at Helsinki University during the beginning of the 90s. He was what can be called a "geek", someone who loves computers and also loves programming. Probably the most significant book, for the creation of Linux, Linus Torsvald has ever read was prof. Andrew Tanenbaum's *Operating System: Design and Implementation*. The book is an explanation of MIMIX system[50]. It represented the closest contact Linus had had with UNIX and it showed him the reliable possibility of creating a Unix-like operating system. In fact, the MINIX is a simple version of an Unix-like system built by prof. Andrew S.

[49] See: http://www.opensource.org/docs/definition.php

[50] *"MINIX is a free UNIX clone that is available with all the source code. Due to its small size, microkernel-based design, and ample documentation, it is well suited to people who want to run a UNIX-like system on their personal computer and learn about how such systems work inside. It is quite feasible for a person unfamiliar with operating system internals to understand nearly the entire system with a few months of use and study."* Reference: http://www.cs.vu.nl/~ast/minix.html

Tanenbaum who decided to develop a minimal system with the same general characteristics of Unix but simpler in order to use it easily as course material. In fact, prof. Tanenbaum was teaching operating systems and he found particularly useful to show the structure of a simple operating system to the students attending his courses. Moreover as a computer scientist, he had been thinking of developing a whole system by himself since his student years when he had meet Thompson and Richies and the Bell Labs. Moreover he was inspired by the idea of developing an entire operating system on his own. Because of his outstanding ability Andrew Tanenbaum succeeded in his purpose, creating MINIX.

Linus was fascinated by the new Unix world opened by MINIX. A radical change occurred in Linus's personal story after this discovery and new Unix-Like ideas did indeed proliferated. He was probably literally inspired by the example of MINIX and so he started the "big project" of his life, following the tradition of many hackers before. It was natural for him to tinker with computer software as well as it was obvious trying to improve them. Despite of the clarifications about the political "coldness" of Mr. Torsvald, the ideological motivations came early in the history of the development of Linux and they found a fertile ground on the Linus "open mentality". In fact in 1991 he went to a seminar held by the famous "longhaired hacker" Richard Stallman at the Polytechnic of Helsinki. Although the meeting did not make Linus an activist because of his natural reluctance towards plain political arguments, he understood that there was something extremely important in the RMS's view and he was conquered by it. As an enthusiast programmer he was interested in the Free Software Project and he recognized the value of many of the features of Free Software. Finally he was keen on supporting the community.

During that period the most challenging task free software and Richard Stallman faced, was the creation of the kernel[51] program of the GNU operating system. The point was to create a well performing kernel able to fit the requirements of the GNU system. Although all the Free Software Community viewed this task as an important work in order to assemble a reliable system, no one was finally able to accomplish it. Richard Stallman had already created almost all the components of the GNU system, but the kernel, the central element, hadn't yet been found. Because of the importance of the topic a vast amount of programmers worldwide were looking for an efficient solution and Linus was one of them, not particularly famous or expert. The difference between his successful project and the stream of previous ones, is likely to be external to the activity of programming, in fact, it was caused by a *process intuition* more than a pure *technical insight*. Linus's major "new idea" was related to the pervasive use of a new medium to foster the software development process, typically the massive use of Web in order to connect programmers worldwide.

[51] *" In computer science the kernel is the fundamental part of an operating system. It is a piece of software responsible for providing secure access to the machine's hardware to various computer programs. Since there are many programs, and access to the hardware is limited, the kernel is also responsible for deciding when and how long a program should be able to make use of a piece of hardware, in a technique called multipexing. Accessing the hardware directly could also be very complex, so kernels usually implement a set of hardware abstractions. These abstractions are a way of hiding the complexity, and providing a clean and uniform interface to the underlying hardware, which makes it easier on application programmers."* See: http://en.wikipedia.org/wiki/Kernel_%28computer_science%29

It is worthwhile to briefly analyze some characteristics of the software development process in order to appreciate the Linus organizational model. This examination is not supposed to be comprehensive of all the determinants that compose the complex practice of writing software nor in any way complete, it only aims at understanding the specific development method used in the case study investigated here. The main advantage Linux had, compared to the previous free-operating system, was the collapse of general cost of creating a reliable technological network to communicate and to share technical ideas due to the invention of the World Wide Web. The possibility to easily connect computers through an already existing infrastructure, the telephone network, made it possible to carry out more efficiently the process of sharing technical proposals about the best way to write the program and, more important, it made the process of fixing software bugs[52] easy. The process of "revision" is one of the most sensitive tasks in software development and usually it requires great organizational effort. In fact, if only one person, or a single team, are able to create the program, a number of reviewers are necessary in order to carry on the long process of revision. In particular the efforts required on behalf of the developer and of the tester are different and this discrepancy can be useful in order to understand Linux success. If a developer spends a "large size effort" to create a new program, the tester can perform well his function, which is basically the delivery of the proper technical feedback, with a "smaller effort" when well supported by the proper communication protocol. Moreover if the cost of organizing a team of developers could be increasingly high due to rising coordination costs, the practice of controlling if the program works properly is less intricate and it requires less creative ability, but more organizational capability. Although it still needs a structured, high sensitive and expensive method to collect technical feedback, it benefits directly from the use of a well performing communicational infrastructure. I think it is sufficient here to point out that the creation of a new software asks for clear specifications of user needs and for a deep comprehension of the inner logic of the problem involved, while the process of revision need only two main steps: a) testing the program that usually has a binary answer as "yes, it works" or "no, it doesn't" and b) the specification of the error. A well-designed communication process, which refers univocally to a precise technical problem, plus a good communication network, which should be large enough to collect many testers as possible, can manage the collection of this data efficiently. The former is central to minimize the technical misunderstandings and to lower the effort needed to comprehend the problem, the latter is necessary in order to minimize the singular effort each testers have to undertake in order to point out the errors. Moreover, it is worthwhile specifying that the use of the software code makes it easier to briefly and efficiently communicate the technical problem because the code is a "univocally described" language with defined/limited semantic variety. This feature makes it easy to find out the problem and to describe it in a simple way.

Therefore this efficient knowledge management empowered by a distributed network of peers, may be seen as the enhancing mechanism of Linux success since it has proved to be the

[52] The term "fixing a bug" means to correct the errors of software programs.

central part of the typical "learning network" which characterizes the open source development process. In fact technical news circulates efficiently through the Net because it is specific and well recognized as well as requiring little time to be addressed and delivered to the right person. The process is empowered by an open network that is committed to link as many testers as possible and that is able to select them in a functional way. Thus these technological features, combined with the distributed topological nature of the network's infrastructure have empowered an effective communication system, which may be seen as the most innovative trait of the Linux development model.

Finally, the most innovative idea of Linus was to use the new powerful communication medium, Internet, as a tool to led a worldwide collaborative work. A few years later he succeeded in developing the Kernel of the GNU/Linux operating system and 15 years later, he became probably one of the most influential persons in the contemporary history of modern computing.

Summing up I would conclude that technology influences the language as McLuhan (1964) clearly has shown but if language influences software so technology influences the way we develop software but if the technology is internet, a distributed open network, so the software that better fits this technology must be open source.

Lessons obtainable from FLOSS history.

The historical overview I addressed here draws important conclusions on the nature of the FLOSS but also it presents methodological problems. In the following paragraphs I will try to briefly account for both.

The first lesson is that the open source phenomenon is not substantially new but it can be considered as the path-dependence evolution of a long lasting tradition in technological knowledge management. It is well rooted in the practices of the Hacker movement and it follows the tradition of *tinkering* with computer and sharing technological knowledge that has always characterized the history of modern computing - and not only of modern computing as the literature on open science (David, 1998) has shown. Nevertheless it developed its own specific technological language, based on the pervasive use of networked media, as long as institutions, the so-called open source licences and business models, based on customization, commodization (O'Reilly, 2005) and communal use and development of software programs, that have proved to be quite effective today.

The second message is that Internet and the World Wide Web have been and probably will be extraordinary important for the phenomenon. In fact the FLOSS community developed a strong ability in using the digital medium and, at the same time, it has been influenced by that use. The link between the medium, the Internet, and the message, the open development model, is highly remarkable in this case, probably the most innovative feature of the entire

phenomenon. Here also the FLOSS has developed its own way of using the medium in order to manage technological knowledge and innovation that seems quite well performing and innovative.

The third lesson comes from the analysis of the birth of free software as the codification of the Hacker development model. The entire concept of free software was so well expressed by Richard Stallman because he already knew what needed to be protected. In fact he understood that the new proprietary model threatened the values of the Hacker technological philosophy that he used to follow. Therefore he codified some essential rules in order to preserve the open model, reacting in an adaptive way to the new technological environment. He founded the GNU project as a practical technological alternative to the proprietary system and created the General Public Licence to protect it. I have shown that the creation of the GNU project as well as the Free Software Foundation have been essentially the transcription of the "Canone Aureo" of the hacker way of developing software - two instruments to preserve his "old-school" technological ideas. In so doing he made the "key variation" of the FLOSS history reacting to the antagonistic paradigm of Windows/PC" and adapting the "old" architecture to new environments. He chose between two alternatives (free vs. proprietary) and he worked on practical solutions to support his choice. Soon other programmers understood the importance of the project and joined the Free Software Foundation. Today almost everyone benefits indirectly from the creation of the GNU/Linux system when Google is used, which is based on big clusters of computer running Linux - as Tim O'Reilly likes to remember speaking about the diffusion of the open source software.

The fourth lesson is that the FLOSS has its own technological story. In this chapter I have only suggested a part of it and the attempt would probably require further perfection but it bears important insights even in such a basic form. In fact, it helps to understand the dynamic change of the phenomenon as it gives chronological order to the large set of facts analysed. Moreover it has been necessary in order to catalogue the complex milieu of ingredients that influenced the development of FLOSS.

Summing up, this historical analysis of the FLOSS technological change has composed a diagnostic picture of the main features of the phenomenon. I tried to account for a) the technological ideas that sequentially influenced the FLOSS way of thinking, b) its evolution and c) its chronological path. Moreover I underlined the strong relation between model of knowledge and system of governance that characterised the evolution of FLOSS, aiming at better understanding not only where it comes from but also how the models of technological communication have influenced its innovation path. As we have seen in the previous pages the relation is clear and powerful because different levels of knowledge appropriability, fungibility and modularity (Antonelli, 2005) have produced different options of knowledge governance that have influenced the creation of technologies, the codification of intellectual property rights and the social participation in the development process.

Finally I would conclude suggesting that as the use of a specific technology influences

language (McLuhan, 1964), which in this case is the programming language committed to writing computer software, which in turn affects technological understanding and technological change, so the open source software seems to fit the requirement of the Internet technology because it favours open communication in a distributed network of agents. As a result if the most important present-day technology is the web, more precisely the WEB 2.0 (O'Reilly, 2005), the open source software would to be its natural way of development.

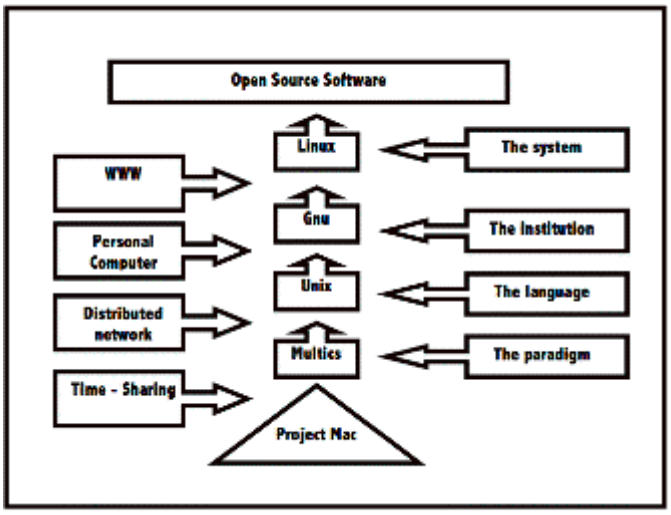

Figure 2: The building blocks of the FLOSS

Methodological remarks

As I obtained the first results from the historical investigation of the FLOSS evolution, I realized that some methodological questions emerged. The whole story needed a proper methodological foundation in order to be a valid explanation and to be useful for further experimentations. So even if I am unwilling to add a theoretical framework to plain facts, since they should be almost self-explaining if I instructed the research properly (Latour, 2004), however it has been necessary to create a structure in order to organize the data collected and to produce the hypothesis addressed previously.

In the same way, although the historical picture makes sense intuitively, some methodological problems arise. They are mainly related to three basic questions: a) how to perform a multidisciplinary analysis of Innovation, b) how to explain and define the evolution of a technology, if there is one and c) how to combine the explanatory power of the macro-analysis with the necessary micro foundations. The problem is complex as a vast literature on Innovation and Technological Change documents (Dosi 1997, Ruttan 1997, Freeman 1994) but here it assumes specific features that are mainly related to three cardinal problems. First of all the comprehension of the open source case demands the account of economics determinants, technological factors, sociological aspects and intellectual property institutions. Secondly the

problem of reconstructing the mechanisms of selection sustaining technological diversity is not obvious and it requires the analysis of the "alternatives" appeared throughout the open source development. Thirdly even if the importance of technological knowledge management suggests that the analysis of the communication process is important to assess the open source evolution, it asks for further localization within limited technological domains in order to be sufficiently effective.

Finally the FLOSS case deserves new explanations (Dalle, David, Ghosh, Steinmueller 2004). It asks for further studies as an open model of knowledge production and its use in different sectors of the knowledge economy, as the biomedical one, is definitely a hot research topic. Furthermore it requests dynamic analysis, which possibly employs dynamic modelling to test the capability of the FLOSS open model to meet the needs of an increasing user base in the future.

In this context, I believe multidisciplinary researches are probably one of the most interesting points of view in order to appreciate, study and probably foresee this dynamic and complex path of change, so it is well worth applying them.

The history of the free/open source software that has been described in this chapter is intended to be one of them.

Maps of operating system evolution

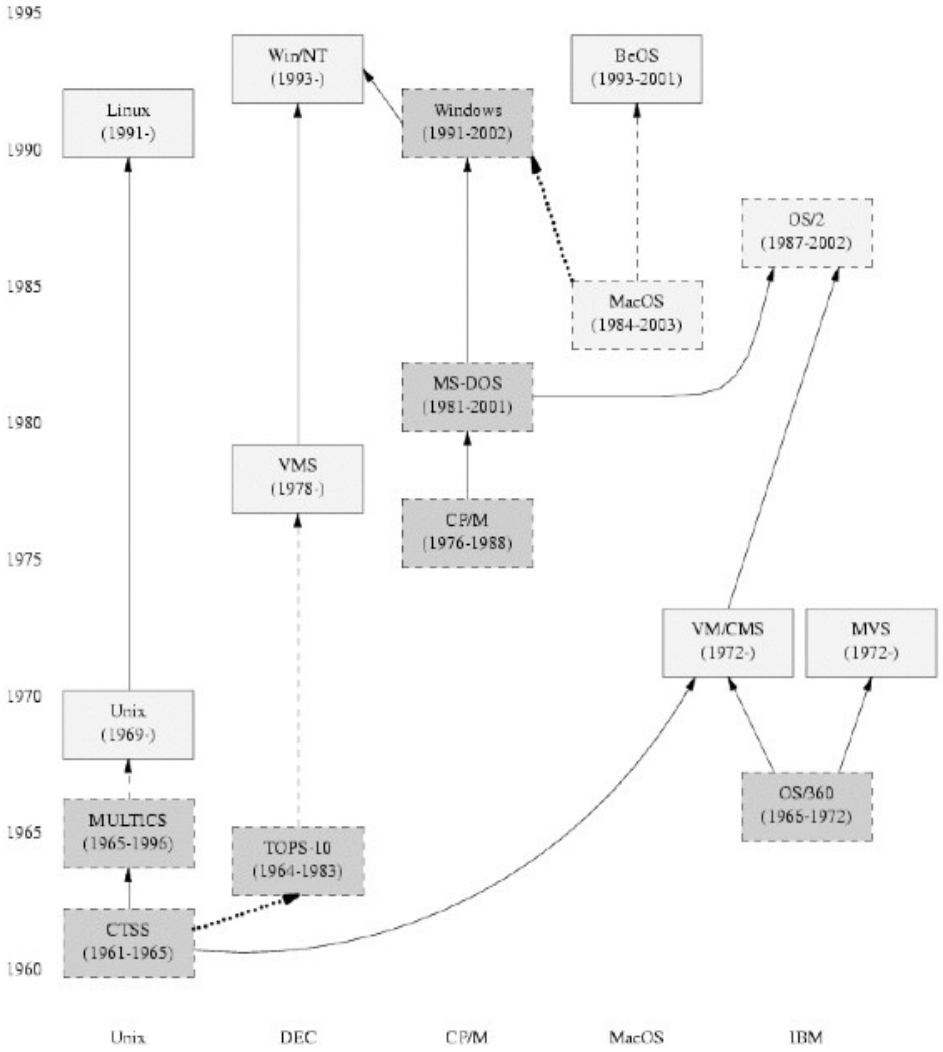

Figure 3 Schematic history of operating systems (Source: Raymond, 2004)

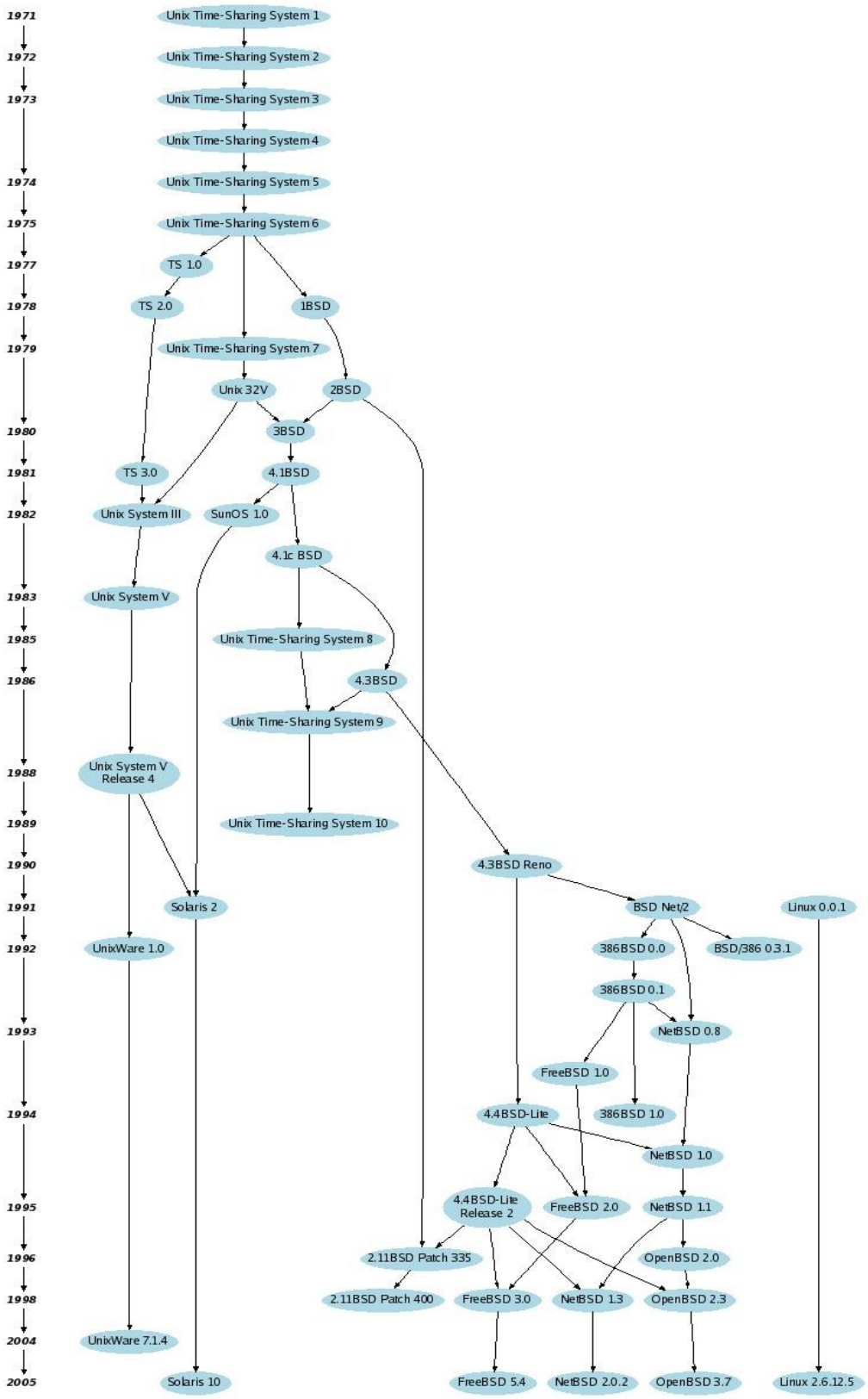

Figure 4: Unix systems (Source, Wikipedia: http://en.wikipedia.org/wiki/Linux)

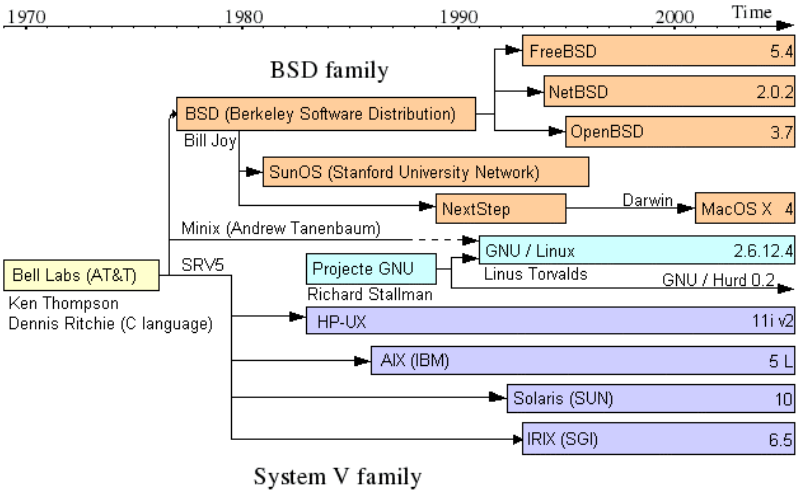

Figure 5. UNIX and BSD family: the "relatives" of GNU/Linux (Source, Wikipedia, BSD)

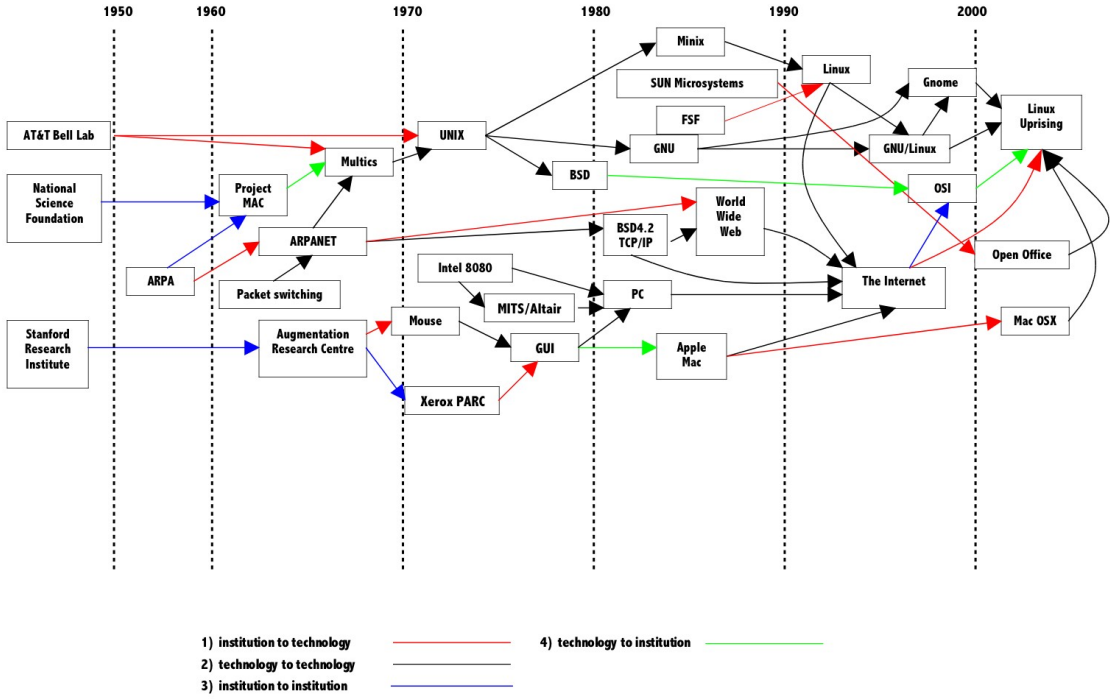

Figure 6 The complex evolution of FLOSS

References

1. Antonelli, C., 2001, *The microeconomics of technological System*, Oxford University Press, Oxford.
2. Antonelli, C., 2003, *Innovation, new technologies and structural change*, Routledge, London.
3. Antonelli, C., 2005, *Models of knowledge and systems of governance*, Journal of Institutional Economics 1 (1).
4. Arthur, W. B., 1999, *Complexity and the Economy*, Science, 284, 107-109.
5. Barabasi, A-L., 2003, *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science and Everyday Life*, Plume books published by the Penguin Group.
6. Baran, P., 1964, *On Distributed Communications,* RAND Corporation Research Documents, vol. I-XI
7. Berners-Lee, T., Hendler and J. Lassila, O., 2001, *The Semantic Web*, Scientific American
8. Barners-Lee, T., 2003, *World Wide Web Consortium: Uniquely Positioned to Lead the Evolution of the World Wide Web*, MIT Laboratory for Computer Science.
9. Benkler, Y., 2004, *Coase's Penguin, or, Linux and The Nature of the Firm*, 112 Yale Law Journal 369.
10. Brooks, F. P. Jr., 1975, *The Mythical Man-Month: Essays on Software Engineering*, Addison-Wesley.
11. Brinch Hansen, P., 2000, "The evolution of operating systems", *Classic Operating Systems: From BatchProcessing to Distributed Systems,* Springer-Verlag.
12. Bush, V., 1945, *As we may think*, Atlantic Monthly, 176, 1, 101-106.
13. Bush, V., 1945, *Science - The endless frontier*, Washington, DC: United States Government Printing Office.
14. Carlsson, B., Holmén, M., Jacobsson and S. Rickne, A. 2002, *Innovation Systems: Analytical and Methodological Issues*, Research Policy, Vol. 31, Issue 2, pp. 233-245.
15. Castells, M., 1996, *The rise of the Network Society*, Blackwell.
16. Ceruzzi, P.E., 2003, *A history of Modern Computing*, The MIT Press.
17. Corbato, F. J. and Merwin-Daggett, M., and Daley, R. C., 1962, *An experimental time-sharing system*, Spring Joint Computer Conference.
18. Corbato F. J. and Vyssotsky V. A, 1965, *Introduction and overview of the Multics system*, Fall Joint Computer Conference
19. Dalle, J.M. and David, P.A., 2003, *The Allocation of Software Development Resources in Open*

Source Production Mode, SIEPR Discussion Paper No. 02-27.

20. Dalle, J.M., David, P. A., Ghosh, A. R. and Steinmueller, W. E., 2004, *Advancing Economic Research on the Free and Open Source Software Mode of Production*, SIEPR Discussion Paper No. 04-03.
21. David, P.A., 1997, *Path dependence and the quest for historical economics: one more chorus of the ballad of QWERTY*, Discussion Papers in Economic and Social History, n. 20.
22. David P.A. (1975), *Technical choice, innovation and economic growth*, Cambridge University Press.
23. David, P. A., 1998, *Common Agency Contracting and the Emergence of "Open Science" Institutions*, American Economic Review, Vol. 88, No. 2.
24. David, P. A., 1999, *A tragedy of the public knowledge commons? Global Science, Intellectual Property and the Digital Technology Boomerang,* SIERP Discussion Paper, n. 00-02.
25. Dibona,C., Cooper, D., Stone, M., 2005, *Open Sources 2.0 - The Continuing Evolution*, O'Reilly Media.
26. Dosi, G., 1982, *Technological paradigms and technological trajectories A suggested interpretation of the determinants and directions of technological change*, Research Policy (11) 147-162.
27. Engelbart, D. C. and English, W. K., 1968, *A research centre for augmenting human intellect*, Fall Joint Computer Conference.
28. Fano, R.M. and Corbato, F.J., 1966, *Time-Sharing on Computers,* Scientific American 215(3):128-140.
29. Foray, D., 2000, *L'économie de la connaissance*, Editions La Decouverte, Paris.
30. Franke, N., e Shah, S., 2001, *How Communities Support Innovative Activities: An Exploration of Assistance and Sharing Among Innovative Users of Sporting Equipment*, Research Policy, vol. 32: 1199-1215.
31. Franke, N. and Von Hippel, E., 2003, *Satisfying heterogeneous user needs via innovation toolkit: the case of Apache security software*, Research Policy 32, 1199-1215.
32. Garzarelli, G., 2002 , *Open Source Software and the Economics of Organization*,ץ in J. Birner and P. Garrouste (eds.), Austrian Perspectives on the New Economy, Routledge.
33. Ghosh, R. A. and David, P. A., 2003,*The nature and composition of the Linux kernel developer community: a dynamic analysis*, SIEPR Discussion Paper.
34. Harhoff, D. H., and Von Hippel, E., 2003, *Profiting from voluntary information spillovers: How users benefit by freely revealing their innovations*, Research Policy vol. 32: 1199-1215.
35. Hayek, F., *The Use of Knowledge in Society*, American Economic Review, XXXV, No. 4;

September 1945, 519-30.

36. Himanen, P., 2001, *The Hacker Ethic : a radical approach to the philosophy of business*, Random House Trade
37. Langlois, R., 1999, *Modularity in Technology, Organization and Society*, Department of Economics Working Paper Series, University of Connecticut
38. Langlois (1992), *Externals Economies and Economic Progress: The Case of the Microcomputer Industry*, The business History Review, Vol. 66, No. 1, High-Technology Industries, 1-50.
39. Latour B., *A dialog on ANT*, version may 2004, http://www.ensmp.fr/~latour/articles/article/090.html
40. Learner, J. and J. Tirol, 2000, *The simple economic of open source*, Working Paper 7600, National Bureau of Economic research
41. Learner, J. and J. Tirol, 2002, *The scope of open source licensing*, Journal of Industrial Economics, 52, 197-234
42. Learner, J. and J. Tirol, 2004, *The economics of technology sharing: open source and beyond*, Working Paper 10956, http://www.nber.org/papers/w10956National Bureau of Economic research.
43. Lessig, L., 2004, *Free Culture How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*, Allen Lane.
44. Levy, S., 1984, *Hackers*, Doubleday.
45. Licklider, J.C.R., 1960, *Man-Computer Symbiosis*, IRE Transactions on Human Factors in Electronics, volume HFE-1, pages 4–11.
46. Licklider, J. C. R., 1968, *The Computer as a Communication Device*, Science and Technology,76, 21-3.
47. McCarthy, J., 1959. *A time-sharing operator program for our projected IBM 709*. Unpublished memorandum to Professor P. M. Morse, MIT, January 1. Reprinted in IEEE Annals of the History of Computing 14, 1, 1992.
48. McLuhan, M., 1964, *Understanding Media: The Extensions of Man*, McGraw-Hil.
49. Metcalfe, J.S. and Ramlogan, R., 2002, *Limits of the Economy of Knowledge and Knowledge of the Economy*, CRIC working paper.
50. Mokyr, J., 1990, *The levers of Riches*, Oxford University Press.
51. Nelson, R., 1995, *Recent Evolutionary Theorizing About Economic Change*, Journal of Economic Literature, Vol. 33, No.1
52. O'Mahony, s., 2003, *Guarding the commons: how community managed software projects protect*

their work, Research Policy 32 (2003) 1179–1198

53. O'Reilly, T., 2005, *What Is Web 2.0 Design Patterns and Business Models for the Next Generation of Software*, Web Article: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
54. Raskin, J., 2000, The humane interface: new directions for designing interactive systems, Addison Wesley.
55. Raymond, E. S., 1999, *The cathedral and the bazaar*, O'Reilly.
56. Raymond, E. S., 2004, *The art of Unix Programming*, Addison-Wesley
57. Ritchie, D.M., 1984, *The Evolution of the UNIX Time-sharing System,* AT&T Bell Labs Technical Journal, vol. 63 no 8, part 2, p. 1578.
58. Salus, P. H., 1994, *A quarter century of UNIX*, Addison-Wesley.
59. Scotchmer, S., 2004, *Innovation and Incentives*, The MIT Press.
60. Stiglitz, J. E., 2002, *Information and the Change in the Paradigm of Economics*, The American Economic Review, 92.
61. Stallman, R., *The GNU Manifesto. What's GNU? Gnu's Not Unix!* (www.gnu.org)
62. Shannon, C. E., 1948, *A Mathematical Theory of Communication*, The Bell System Technical Journal, vol. 27: 379–423.
63. Shapiro, C., and Varial, H. R., 1999, Information Rules: A Strategic Guide to the Network Economy, Harvard Business School Press.
64. Swann, P., 1997, *Personal Computer Part One*, Case study - Manchester Business School teaching material.
65. Torsvald, L. and D. Diamond, 2002, *Just For Fun: the story of accidental revolutionary*, Texere.
66. Utterback, J. M., 1994, *Mastering the Dynamics of Innovation*, HBS
67. Von Krogh, G., Spaeth, S. and Lakhani, K., 2003, *Community, joining, and specialization in open source software innovation: a case study*, Research Policy, vol. 32, 1199-1215.
68. Weimberg, G.M.,1971, The psychology of computer programming, Van Nostrahnd Reinhold.
69. Zeitlyn, D., 2003, *Gift economies in the development of open source software: anthropological reflections*, Research Policy, vol. 32: 1243-1258