Open source as social organization of production and as a form of technological innovation based on a new conception of property rights

Presentation by Manuel Castells in the World Social Forum 2005

Open source refers to a form of social organization of production that originated in the development of computer sofware, and it is mainly concerned with the open access to the knowledge of the source code of a software program. A software code is a set of instructions for a computer. The source code is a list of instructions that constitute the fundamental formula for a software package. It is the DNA of a software program. Most commercial software is released in machine language, in binary language that machines can understand, but not humans. The source code is the formula for this binary language, and with the source code it is possible to understand the logic of the program, and thus, anyone with the technical knowledge can modify it. It is organized around a special notion of property. Most commercial software, such as Microsoft´s, is based on the control of the property rights of the source code. Software programs can be sold to the users, and they have to accept what they receive because they cannot access the logic of the program, so they cannot modify it, given the fact that they are excluded from the knowledge of the source code. Conventionally, in a capitalist economy, property is the right to exclude others from the use of a good or service. In open source, property is configured fundamentally around the right to distribute, not to exclude. The source code for open source is published and distributed for the use by anyone who wishes. And because the source code is known, users can modify it, and can modify or generate new applications. Free source code is open, public, and non proprietary. This new form of property, that is entirely contradictory with the usual regime of intellectual property rights, is supported by a governance system that holds together a community of producers. It is based on human motivation to work within this logic and is supported by an evolving set of organizational structures to coordinate behavior.

Open Source is not a fantasy or a marginal practice. Very large, and very important software development projects have resulted from an open source process of production. The best known are Linux and Apache, but there are many others, and this is an expanding practice in the research world, in the hackers world, in the education world, in the institutional world and in the business world, including some large corporations, such as IBM. As of 2004, Linux is the operating system for more than a third of active web servers in the world, and it is the operating system for about 14% of the large server market. Apache largely dominates the web server market - with 65% of all active web sites running Apache. Sendmail, and many other popular programs, are also produced and distributed as Open Source. Open source introduces a new, cooperative form of production that transcends the traditional limits of the social division of labor built on hierarchies. Indeed, open source works as an open network of voluntary cooperation.

Although Open Source did not start on the Internet, the Internet is a fundamental platform for the scalability and interactivity of the cooperation process at the base of Open Source.

Open Source is not necessarily anti-capitalist. There are many capitalist firms, including very large corporate firms that practice open source. But it is a-capitalist, meaning that Open Source is compatible with different social logics and values. It does not need the incentive of profit to work, and does not rely on the private appropriation of the exclusive right to use and enjoy the product. It is based on a form of social organization that has profound political implications and may affect the way we think about the need to preserve capitalist institutions and hierarchies of production to manage the requirements of a complex world.

The context of open source

The context that surrounds the development of open source, as a social phenomenon, a political phenomenon, and an economic phenomenon includes at least four major features:

1) The Internet transforms the nature of the process of work, enhancing interactivity and distribution. Network organization becomes effective, particularly with increased telecommunications bandwidth. In the open source form of production as important as the code itself is the process by which it is built.

2) Open source expresses the development of new relationships between community, culture, and commercial activity. The open source community is based on a set of rules and shared values. In addition, from this cultural autonomy the community relates to the rules of the capitalist organization that characterizes the broader context. In fact, as in the history of industrial organization, ideas create institutions, that set up production processes. Thus, the ideas behind open source are at the roots of a new logic of production.

3) Open surce exposes the new logic of organization of production in a knowledge intensive economic process. The development of software is made up of digitally encoded knowledge that combines from the bottom up in the process of production. Furthermore, as mentioned above, open source is an experiment in production built around a distinctive notion of property. The traditional notion of property is based on the right to exclude the non owners from the use of something that is owned by someone. On the other hand, open source property is configured around the right to distribute, not the right to exclude. This is in fact in the tradition of "fair use" of intellectual products that are used without securing their property. Under an extended notion of fair use, no individual´ s fair use will be permitted to constrain subsequent fair use by another individual and for any purpose. (On "fair use", and the transformation of the notion of intellectual property rights see the definitive analysis by Lawrence Lessig "Free Culture”, 2004)

4) Open source is a broad social phenomenon, not limited to the field of software, but applicable to the production and distribution of knowledge in a variety of domains.

History of the Open Source Movement/Practice

In the early stages of the digital industry, there was no distinction between hardware and software. The producers and users of each machine had to write the software necessary to interact with the machine. This ad hoc language could not be used for other machines. Then, in 1956, in the US, the Department of Justice, under the anti-trust law, decided that Western Electric and ATT could not engage in manufacturing other than related to telecommunications. ATT lawyers, in a cautionary strategy, promoted the licensing of software and other communications technology at nominal fee, so not to be accused of producing software (a non telephone good) for profit. This decision allowed the transfer from the software research conducted at Bell Labs into the public domain. In 1964, researchers at MIT, in cooperation with Bell Labs and General Electric, developed a time-sharing system called Multics. The software component of the project was difficult to implement, and in 1969 Bell Labs withdrew from it. But two researchers at Bell Labs, Ken Thompson and Dennis Ritchie, decided to continue on their own. In the summer of 1969, they created an operating system kernel (the heart of the program) that named UNICS (a joke to deride MULTICS), later to be renamed UNIX. UNIX is today the main operating system for the Internet, as well as for researchers and for major business software packages. In October 1973, Thompson and Ritchie presented a paper on Unix at the symposium of ACM. From then on, some of the most innovative programmers communities showed their interest in UNIX. Because of perceived legal constraints, ATT-Bell Labs licensed UNIX upon request, first to universities and researchers, then later to military and commercial organizations. The software was communicated without technical support. Thus, users started to improve the program by themselves. ATT supplied, for a few hundred dollars, the source code written in the programming language C. Unix could run on any machine that had a C compiler, not just on the DEC machines on which it had been developed. UNIX provided a most useful teaching tool that could be understood and modified because of the availability of the source code. It became very popular in computer science departments and diffused around the world. Since the Internet was still very limited, the communication of results from UNIX developers took place in meetings and seminars, that created an international community of UNIX users. One of the earliest computer science departments to adopt UNIX was Berkeley. In the Fall of 1975, two professors Stonebraker and Fabry could buy a new, powerful computer, a PDP-11/70 to run on UNIX. At the same time, Ken Thompson arrived in Berkeley to spend a sabbatical year. And a new cohort of graduate students entered the department, including Bill Joy and Chuck Haley. With the help from Thompson, by theSummer of 1976, Joy and Haley, working on a Pascal system, were able to improve the UNIX kernel. As news spread around the Unix community, Joy put together a package of tools and utilities that he named the Berkeley Software Distribution (BSD). From then on, BSD improved in several versions, through the collaboration of Berkeley students and other UNIX programmers, around the world, including a group of teenagers in a high school in Boston, that made major improvements on version 7 of BSD, allowing the program to be ported to a number of different machines. At this point, and with great demand on UNIX, ATT started to think about controlling the product, so they just restricted the use of their license to one university (Berkeley) and only for teaching and research purposes. By the end of 1979, Joy was able to release 3BSD, a complete UNIX operating system distribution running on the powerful VAX machine. This was a major factor behind for the decision of DARPA (the research agency of the´ Pentagon, and the sponsor of ARPANET) to utilize UNIX to run its computer networks. Arpa funded Professor Fabry to set up a research group in Berkeley, the Computer Systems Research Group (CSRG), that released 4BSD in 1980 and later, in 1983, 4.2BSD, a system able to integrate TCP/IP networking (the Internet protocols of communication) into UNIX. 4.2BSD is the software foundation of the Internet as we know it today. But in 1984, ATT and Bell Labs were separated by a judicial decision, and Bell Labs was allowed to commercialize its research. Then Bell Labs created a Unix System Laboratory and started to license Unix at the ` price of hundreds of thousands of dollars. From then on, BSD and ATT-UNIX took separate paths, and ATT engaged in a series of litigations to restrict the free diffusion of BSD, and of UNIX in general.

As a reaction to the attempt by Bell Labs to commercialize and restrict the use of UNIX, a group of programmers at MIT´s Artificial Intelligence Lab, led by Richard Stallman, created in 1984 the Free Software Foundation to build an operating system that while based on the UNIX tradition would not be UNIX and could be freely distributed and used. They called it GNU (for GNU´s Not Unix). Stallman published a GNU Manifesto defining free in the following way (saying it was free as in "free speech", not as in "free beer"): a) Freedom to run the program for any purpose b) Freedom to study the program and modify according to needs c) freedom to redistribute copies, either for payment or without payment d) freedom to change the program and to redistribute the program in the public, so that everybody can do the same. To enjoy these freedoms users would need unrestricted access to the source code of the program. Stallman created a legal/institutional tool to enforce this freedom: the General Public License (GPL). Under this license, Stallman substituted "copyleft" for copyright. Software licensed under GPL, and its derivatives, cannot be made proprietary. Furthermore, GPL code cannot be used in any combination with proprietary software, unless the entire combination is released under GPL (the so called "viral clause"). The Free Sotware Foundation has created some very good software, particularly the GCC compiler, and the GNU Emacs text editor. However, the resources of the Foundation were limited, and they were never able to build a full operating system under GPL, so that GPL and the culture it manifested became more important than the software produced by Stallman and his collaborators. This was mainly because Stallman worked with a small groups of programmers, under close intellectual control, and did not use the cooperative power of the Internet. Thus, by the late 1980s, GPL/GNU was a limited movement, restricted to small networks of programmers.

In the meantime, BSD had a number of commercial applications (Sun Microsystems being the most notable), but the original research group in Berkeley (CRSG) became entangled in a series of disruptive litigations between ATT and the University of California, so that by 1995 the research group was disbanded. In the corporate world, a number of major Unix users (including among others

IBM, HP, Siemens and others) created an Open Software Foundation to counter ATT´ sattempt to control UNIX. But the crisis of the computer industry in 1991, led to the demise of this foundation. By the early 1990s, it looked like the idea of open source software was limited to countercultural programmers without the tools to implement their dreams. Microsoft appeared on the edge of achieving quasi monopoly on operating systems software, and UNIX and its derivatives were being entirely privatized. Then, two major events reversed the trend: Linux, and the expansion of the Internet.

Linux originated from the needs of Linus Torvalds, a 21 years old computer science student at the University of Helsinki. In 1991, working on his master thesis, he wanted to use UNIX, but without access to a powerful machine, he wanted to adapt UNIX to his new PC 386. He felt he needed much help. He started by writing a very primitive kernel for his program (The kernel of an operating system is the base layer of instructions that control the key information processing and resource allocation functions that make a computer work).

In the Fall of 1991 Torvalds released Linux version 0.02 (He called it Freix, but the systems administrator who posted the program renamed it as Linux). Torvalds posted his kernel on the Internet, and asked for help to develop it. Naturally, he provided the source code, and explicitly authorized modification of it, expecting that the modifications will be released so that the network of interested people could benefit from everybody´ swork. Given the growing interest in the program, in January 1992, Torvalds released a new version (Linux 0.12) under a GPL license, including the "viral clause", so that all the contribution to the development of Linux would be open and free. From the very beginning, Linux developed through a network of cooperation that debates openly on the Internet, and that, while engaging in harsh technical debates, never paid too much attention to ideological disputes. Besides, having born free, it never had to litigate in its origin against corporate lawyers (only in 2002 started some vicious maneuvers inspired by Microsoft to pretend that Linux had appropriated software from other companies - without much success in the courts up to 2004). So, Linux rapidly grew to a network of thousands of contributors, with a core of a few hundreds of programmers, and an estimated 21 million users by 2002. In spite of the size of the network and the growing complexity of the program, Linux avoided "forking", that is the development by a segment of the community, of a different version of the code that would be incompatible, thus dividing the cooperative efforts (see below). This was largely due to the opennes by Torvalds to accept substantial modifications of the code, and also to let other people to propose and control key components of the code. This was particularly the case of the effort to develop an operating system for the Internet, without using code that was tied up in the BSD dispute between ATT and Berkeley. The effort led by English developer Alan Cox, was acknowledged by Linus Torvalds and integrated into the program.

As we will see below, Linux has been gradually accepted by major corporations, governments, and institutions, as well as by a growing number of businesses that develop Linux applications and services. All this without losing its character of an entirely open source program, without private appropriation of its software, and without profit making on the part of Linus Torvalds or anyone of the major developers in the network. And Linux continues to be developed and maintained by a global community interacting over the Internet without formal hierarchy or assigned division of labor.

The other major experiment in Open Source is the Apache web server program. A web server program is the software that answers a query by returning a web page to the computer that asked for it. The first web server was daemon, developed by Rob Mc Cool at the National Center for Supercomputing Applications at the University of Illinois. When Mc Cool left the Center in 1994, the management of the Center did not continue the work. The work was continued in 1995 by Brian Behlendorf, a Berkeley student, working on the web publication of Wired Magazine, and member of Cyborganics, a countercultural, virtual/physical community in San Francisco. Because the NCSA code was open source, Behlendorf could work on it. With eight other developers they started a project. Three months later, there were 150 suscribers to their project´s mailing list. In December 1995 they released Apache 1.0. Because many of the developers were involved in commercial web site development activities, they released the code under BSD license, not GPL: The key difference is that BSD requires to release the source code only for those parts of the code that have been originated from open source contributions. But it does not preclude the mixing with propiretary software for commercial use, on the condition of still preserving open access to the information that was originally released in open format (that is: it does not contain the "viral clause"). The Apache group is organized in a network of contributors, with an elected, rotating committee, and procedures for voting and election. It has a sort of internal constitution. Apache continues to develop, and improve, always as an open source project. In 2004 it was the program that run over 2/3 of the web sites of the world wide web. It has spurred a considerable number of businesses, including much of Internet-related business for IBM.

From free software to open source

Linux 2.0, released in June 1996, marked the maturity of the technology. It also expressed the departure from the ideological stand taken by the Free Software Foundation. The Linux community was, by and large, not interested in undoing capitalism or challenging coventional property rights. The common purpose was to develop good software, and make sure that the conditions of free access to the source code would be respected because that was the key for good quality software. Besides, a growing number of business-oriented people, such as Tom O´Reilly, were trying to make compatible the freedom of knowledge with the business applications of Linux and other programs originated by the free community of developers. Eric Raymond, the intellectual spoke person of the free software movement, author of the de facto manifesto of this culture (a book published in 1999 under the title "The Cathedral and the Bazaar") also supported the strategy. In a meeting held in the company VA Linux in February 1998, with Raymond, O´ Reilly, and the CEOs of VA Linux and Red Hat (another Linux-based company) they proposed the word "Open Source", that was endorsed by the free sotware summit in April 1998. The new Open Source definition was based on GPL, but also could incorporate other forms of license, inspired by the practice of another open source company, Debian. BSD could also be accommodated under the new definition.

The definition of open source, as a result of this process, says that "the program must include the source code and must allow distribution in the source code as well as compiled form". Concerning derived works: "the license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software". It is important to observe that the statement says "allow" and not "require", unlike GPL. BSD, on the contrary allows distribution of free of open source, but also allows that the code would not freely distributed. This means, that other users may combine free software and proprietary software according to their needs. The compromise on the principles allowed the open source movement to expand into the mainstream of social and business practice.

Open Source and Business Models For a long time, the world of software evolved along two largely incompatible lines. Free sofware developed by voluntary communities of programmers, and proprietary software created by business firms either for their in-house use or for sale. With the commercialization of some versions of BSD and, later, with the success of Linux, the boundaries between the two worlds of software became blurred. Certainly, Microsoft sticked to its monopoly of inferior technology imposed by aggressive business practices that several times ended up being challenged and sentenced in courts, or subject to hundreds of millions in fines, as the one imposed by the European Commission. But a substantial part of the information technology business world found commercial applications in process and in product by adopting open source. Some companies, such as VA Linux and Red Hat, distributed Linux respecting the open source rules, but made money out of their packaging, as well as from instructions for use and support services. Then, from 1998 onwards a number of major companies started to use Linux as the basis of their software packages for business, always respecting the rules of open source. This was the case of Oracle (using both Linux and FreeBSD), Computer Associates, Informix, SAP, Hewlett Packard, Dell, Compaq, Silicon Graphics, Gateway, and, the most noticed convert, IBM, together with a long list of business firms. In 2000, a major company could use Red Hat Linux on a Dell server, running on an Oracle database, with global support services from IBM.

Furthermore, some companies started to release their source code. This was the case of Netscape, releasing the source code of Communicator 5.0 in January 1998, to counter the unfair competition from Microsoft´s Internet Explorer. However, in order to keep some commercial control, Netscape released the source code under two licenses, Netscape Public License, with some restrictions, and Mozilla Public License, similar to GPL. The existing restrictions and the lack of clarity in the relationship between the two licenses, made the sofware community very reluctant to cooperate with the development of the code under the Mozilla license. The attempt was, by and large, a failure. On the other hand, IBM used a different strategy to develop its web server program. After rejecting the acquisition of Netscape, and a partnership with Microsoft, IBM decided that Apache (remember: an open source program developed by a cooperative of programmers), was the best, and most diffused, server program. But instead of appropriating it, or developing an IBM program on the basis of open source, IBM decided to join Apache, rather than Apache joining IBM. IBM became a member of Apache, with a seat in the Apache committee, and would contribute its huge resources to the development of the program, and release all its contributions to the Apache code, as open source. At the same time, IBM started to provide support services to companies using the Apache sofware in their web services, which helped considerably in the development of Apache in the business world. On December 1998, Lou Gerstner, the visionary president of IBM, decided to support open source software, and Linux, as part of the core strategy of the company. To show its commitment to open source, IBM released its own source code, based on Linux, to run one of its major mainframe computers, System 390.

Even Microsoft saw the writing in the world. In August 1998, Microsfot´s high level executive Vinod Valloppillil wrote a confidential document that, after it was diffused, became known as "the Halloween Documents". In this memo he acknowledged the quality of Linux software and the effectiveness of open source as a method for programming, and identified it as a direct threat to Microsoft. He wrote: "Linux and other Open Source Software advocates are making a progressively more credible argument that OSS software is at least as robust - if not more - than commercial alternatives". Moreover, according to the Microsoft document: "The intrinsic parallelism and free idea exchange on OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mind-share threat" and "the ability to collect and harness the collective IQ of thousands of individuals across the Internet is simply amazing. More importantly, OSS evangelization scales with the size of the Internet much faster than our own evangelization efforts appear to scale". It cannot be better expressed the logic and potential superiority of open source development. Naturally, Microsoft wanted to fight against Open Source, but its traditional monopolistic (and not always legal) practices were against companies, and not applicable to fight open source. Indeed, the Halloween documents acknowledged that for Microsoft "to understand how to compete against OSS, we must target a process rather than a company". The ultimate software business, Microsoft, the most succesful company in the world, was on the defensive when confronted to open source.

Indeed, the open source model undercuts the conventional business logic. Power in this market shifts away from software suppliers and toward software customers. But there is still possible to combine business (money making) and open source. In different ways:

Generic business models by commercializing useful packages of open source programs and their applications, with or without proprietary programs. This may include technical support systems, services, and training. This is particularly the case of VA Linux and Red Hat, or Bitkeeper, or Hewlett Packard giving away the code source as a way to popularize a program. This was the case of Netscape (see above) to build a proprietary system on top of open source programs. This is the case of one of the latest Apple Computer´ sdesktop operating system, OS X, built on two open source systems. The strategy behind is that application developers switch from writing for Windows to writing for MAC OS, because they have access to the source, and confident that the market will move into this direction. To use the potential of the worldwide community of developers as a way to enhance the technical edge of the company´s products. This is the strategy of IBM (see above) that offers all its major enterprise applications on the basis of Linux platform. In addition, the company also offers web services, based on open source, to compete with Microsoft´s .net releasing the source code with some restrictions to benefit from the community support but also imposing some conditions in favor of the company. This has been the case of Sun Microsystems with Java and then Solaris. The Open Source community has been ambivalent vis a vis Sun´ sand similar strategies.

But perhaps the most critical business development of open source is to change the organization of work, and the production relationships. We will examine this point later, after we understand better how open source works.

How open source works

Open source is a knowledge production process undertaken by a community that has harnessed the communicative and collaborative power of the Internet.

Open source raises four challenges in contrast with the usual form of organization of production in a capitalist economy:

a) The motivation of the individuals. Why skillful programmers contribute their time and effort without compensation?

b) What is the economic logic that departs from conventional market logic?

c) Coordination. How hundreds of individuals cooperate freely in a project without a central hierarchy that organizes the division of labor. How coordination is implemented outside market mechanisms of hierarchical decision making?

d) Management of complexity. The development of software is a highly complex endeavor that is not solved simply by adding manpower. In fact, the classic study by Frederick Brooks shows that increasing the number of programmers increases the problems in successfully completing the program. This is because with an increased number of programmers, the work that gets done scales linearly, while complexity of the process and vulnerability to mistakes scales geometrically. Under such conditions, the question is: what is the procedure of governance that enables the community of programmers to achieve the quality of the expected program in such a complex process of work?

Let us take these questions in sequence.

Individual Motivations

We have some surveys on the motivations of Linux developers, although certainly not statistically representative of the community. The picture that emerges is that the typical Linux developer is a person who feels part of a technical community, who wants to improve his programming skills, to benefit from better software, and wants to have fun.

He does not care about money rewards. In fact, he cares more about the time he has to spend in task. Main motivations are: individual learning, work efficiency, and having fun. In a 2001 survey conducted by Boston Consulting Group, the responses from Linux developers allowed to typologize them in four groups:

i.The believers. Motivated by their conviction that software should be open (1/3 of respondents).

ii.The professionals, who use open source because helps them in their jobs (1/5 of respondents)

iii.Fun-seekers, who do it for intellectual stimulation (1/4)

iv.Skill-enhancers, for whom open source allows them to become better programmers (1/5)

In addition, most observers, including the path breaking book by Eric Raymond, insist on the importance of reputation among the peers of the community in motivating the programmers. This is not too different from the scholarly community. To be recognized by the peers that you respect is one of the highest rewards. Also, an important motivating factor is the belief in the benefits of the community, in the goodness of cooperation. The belief is that the community empowers the individual to help himself. It is a sort of communal individualism. And last, but not least, the belief in innovation and experimentation as the highest, most valued form of human behavior.

The economic logic of a collective good

Eric Raymond likens open source to the gift economy of some societies. Gifts are source of prestige, status, and self-esteem. This is one part of the economic logic behind Open Source. But there is an additional economic logic. Open source software is a nonexcludable good because no one is prevented from consuming it. It is also entirely nonrival in the sense that any number of users can download it and use it. So, this is essentially a public good. All public goods are submitted to the danger of the free rider attitude, as theorized by Mancur Olson. This means that there are people (free-riders) who can take advantage of the common good and do not contribute to it. It is like eating from the common pot in which you do not put anything. So, if a high proportion of people take this attitude there is nothing in the pot. However, if you feel compensated by the act of creation, even if other people take advantage of it, as long as there are enough contributors, you do not care about parasites, because in the Internet world, the marginal utility of an additional copy of a software program to me is zero. I, and my collaborators, do the work, and are rewarded by our creation. If others just consume it, this does not alter the joy of my (and our creation) on the condition of having enough contributors to have cool software. Rather than a non-rival good Open Source software is an anti-rival good, a network good, that creates value out of networked cooperation.

The method of coordination

The key in the process of cooperation is freedom to join and freedom to quit. Indeed, the right to forking is openly recognized in any open source community. Forking means that at one point, anyone may decide that he does not agree with the decisions taken about the code, and starts a different programming line that, ultimately, becomes incompatible with the software that is being developed in the community. Linux has been close to forking in some instances. Forking is damaging to the community because it disperses efforts and resources. But it is essential for participants to know that they can always propose their own alternative, if they reach a crisis point. But, how coordination is achieved?

First of all, because of cultural norms in the community. This is made concrete in the right of ownership to the code, meaning the right to distribute modified versions of software. Ownership of this kind is acquired by initiating the program, (the founder), by receiving explicitly the ownership from the original founder, and by picking up a project that seems to be abandoned.

Second, decision making mechanisms imply a certain form of accepted hierarchy, with the founder, his recognized lieutenants for certain lines of development of the project, a core of maintainers of the program, and a broader group of credited developers (publicly credited in the mailing list of the project). This hierarchy develops largely spontaneously and it is essentially founded on the recognition of the technical capacity of the cooperators. So, at the heart of open source there is a matter of technical rationality. However, the ability to keep the community growing and cooperating it is also related to the leadership. The leader of a project must be charismatic and respected. But, at the same time, he must respect all the contributors, and accept criticism, both technical and personal. The exchanges in the community can be very harsh, but ultimately it comes down to respect the right of everyone - or else, forking. The important matter in exercising leadership is that all communications are public, so that the community can always make its own opinion. The most important contribution of Linus Torvalds was not his original kernel, but his ability to build a community of cooperation over the Internet. The main crisis for Linux was when in 1996, Linus moved to Silicon Valley, to work on Transmeta, and at the same time, was taking care of his daughters. He became overwhelmed with responsibility and could not answer all the requests of the community. The criticism became that Linus (not Linux) does not scale... The leadership response, after some bitter exchanges, was to decentralize decision making on some parts of the code, although retaining final word on technical controversies, to preserve the unity of Linux, and its cumulative development. So, leadership combines benevolent dictatorship with decentralized decision making and with embedded interest in cooperation from the whole community as the only way to achieve the goals in writing good software. There are egos certainly involved in the process, and this is normal, and accepted. What is not accepted is to use a power position for personal benefit, meaning appropriating others´ work and not release or contribute. If some people make money on the side, or publish, or whatever, it is OK, on the condition of not keeping for themselves any information or innovation that is collectively produced by the community. This is not about a community of angels or countercultural activists. This is about a cooperative of technical programmers that know this works better than proprietary software, and, in addition, they enjoy the chance to innovate and to be appreciated and recognized by their peers.

The management of complexity of software development

Sofware is a very complex product. The Red Hat version of Linux 7.1 (which includes some applications) has over 30 million lines of code. In formal organizations, the process of production for such a gigantic task would imply a very complicated division of labor with huge numbers of programmers to be coordinated, inducing massive problems of efficiency (remember Brooks´ analysis). This is in fact the case in Microsoft, and this is why Microsoft´ ssoftware is full of bugs and mistakes (the blue screen of doom, as the free programmers say) the key to reduce the problems posed by complexity is modularization, meaning to differentiate the task by subsets of problems to be solved while keeping the compatibility of the software. But it is difficult to plan in advance who should do what since this is a process of innovation. It is through trial and error that the program advances. Open source cooperation over the Internet makes transparent the progress of the program and the lines of development adopted by the subsets of the community concentrating on some of the modules. Indeed, each group of the community has the responsibility assignment of solving a problem, rather than executing a preplanned task.

Management of cooperation

The way to ensure discipline cooperation is through the sanctioning process of public blame (or flaming) against members of the community that do not abide by the implicit and explicity norms and rules. But, something else is needed beyond a certain level of complexity: a formal governance structure for large scale projects. This varies with projects. For instance, Apache, that started with 8 people in 1995, grew quickly to a core of several dozen developers dispersed around the world. Then, hundreds of contributors came on board. Thus, Apache organized a political system. Anyone can express an opinion. But the only binding decisions are those taken by the Apache group. Members are elected to the Apache group on the basis of their technical merit and their contribution to the program, on the basis of peer review (not too different from academia). There is veto power on decisions on program: any change in the code requires at least 3 positive votes and no negative votes. But any negative vote must be motivated. In 1999 the Apache Group adopted a legal structure as the Apache Software Foundation, that provided an institutional umbrella to web-related open source projects, such as Jakarta, Perl, TCL etc.

Linux has a semi-formal hierarchy. At the top, naturally, there is Linus Torvalds. But around him, originally, where the "elders of the tribe" (usually younger than 30 at the time) coopted by Linux on the basis of their contribution to the code. In 1994 Linus released a list of 80 developers in an official credits file. In 1996 he added a maintainers list. A maintainer is reponsible for one specific segment of the code. Later on, Linus designated some lieutenants in charge of broader responsibilities. Formal recognition of status in these lists helps people with their reputation, and eventually with jobs and offers outside the Linux community. In exchange, they respect the organization of the distributed work. Some times, the excessive centralization of decision in the hands of Linux becomes dysfunctional and leads to revolt, as in 2002. But replacing Linus as the last authority would actually disorganize the cooperative network and stopped the development of Linux, so ultimately some form of arrangement (in this case a "deputy pinguin" to Linus - penguin is the mascotte of Linux) sorts out the problems, in a constant problem of adjustment. At any rate, between 1991 and 2005, Linux has evolved with hundreds of core developers cooperating in the program, and thousands of occassional contributors, without major disruption and without excessively formalized organization.

Open source as cooperative organization of production

Open source is a way of organizing production, challenging traditional forms of division of labor, organizational hierarchy, and conventional property rights. A number of analyses propose the possibility of extending this form of organization to many other areas beyond software, on the basis of the principles that characterize the open source process: User-driven innovation that takes place in a parallel distributed setting cooperative behaviour regulated by cultural norms and governance rules the economic logic of non excludable, anti-rival goods, and network externalities and synergies. A redefinition of the notion of property rights. Property rights in open source are built on the right to distribute, not to exclude. Remember that property is a socially constructed notion. The experience of intellectual property rights in the music distributed over the Internet is an important illustration of this new principle, and also of its contradictions. The Internet allows and enhances this new system of cooperation, while creating serious difficulties for the enforcement of traditional property rights.

The generalization of open source to other domains of activity, is based on the implementation of four principles:

a) Empower people to experiment, and provide them with the appropriate technology, and the required social incentives

b)Find an engineering solution for bits of information to find each other

c)Structure information so it can recombine with other pieces of information (modularization)

d) Create a governance system that sustains the process (the GPL logic is an example of institutionalization of new property rights)

Open Source, technology, and world development

Because open source challenges oligopolistic uses of intellectual property rights (including technology) it encourages diffusion of knowledge, and innovation in the applications of existing knowledge, adapted to the needs of the users. This is why Open Source has prompted hope in many developing countries where programmers can develop their skills by creating programs geared toward their needs without being constrained by the property rights of multinational corporations. Furthermore, by incorporating in the cooperative networks a great number of programmers from the developing world, and by sharpening their skills in the community, the whole world will unleash a much greater innovation power. This is why countries and governments all over the developing world (particularly in Brazil, but also in China, India etc) are adopting open source and experimenting with its potential. Example is the Simputer project in India, a $ US 250 portable computer running Linux, designed by the Indian Institute of Science and the Bangalore company Encore. China also has developed Red Flag Linux and other development packages. Certainly Microsoft is launching a major campaign in China and other countries, often seducing public officials, to counter the spread of open source. But the diffusion of open source is rooted in the development needs of countries and organizations around the world. The applications of open source in software and computing power could have dramatic effects in health and education in the developing world, for instance in primary care. Therefore, the ideological and commercial battle around open source is a major issue in development nowadays.