FREE SOFTWARE AND THE INFORMATIC WORK IN KNOWLEDGE'S CAPITALISM

Pablo Miguez
University of Buenos Aires, Argentina

Abstract.- In this paper, I try to analyze the consequences of the inmaterial work in the software production, within the digital citizenship. I study relationships between informatic workers within software and informatic services production, comparing two models of software development. One inspired in free software and the other based on the propietary one. The informatic workers labour has been analyzed looking at previous theories about the free software movements. Some of these theories use concepts from the economical anthropology in a singular way. This way is questioned when we look at the organization of the informatic work. The informatic work is the software development, this is the construction of programmes as final products; where teams, projects and networks coexist. The labour process had important changes when informatic technologies were inserted. These changes have created a new kind of digital citizenship, but explotation has not dissapear. It has acommodated to new circumstances.

Free software and the informatic work in knowledge´s capitalism

In this paper I try to see the relationships between informatic workers within the process of software production and informatic services. To do that, I compare between two software development models, one inspired in free software, and other that comes from the propietary software.

Until the 80's the informatic revolution was only seen in US, in the universities and laboratories. In the 90's the informatic revolution was spread all over the world, accelerated by the Internet consolidation.The "knowledge society theories" started to stay at the top of the academic agenda.The free software communities are shaped by lot of innovative users.

While the propietary software model of Microsoft is based in the "new" knowledge "enclosures", the free software is an open and cooperative technoinstitutional model. In this paper, we will analize these two models. We'll see the free software case to show that some of the hegemonic theories within the field, use to take some categories from the economical anthropology to become legitimated. In the propietary software case, we'll show the inconsistences between the theories that use to apotheosize the new types of job; and we will show in which way the etnographic view is pertinent to analize the practices and meanings of the informatic workers.

The Free Software Movement

Some different papers about free software started to be frecuently made since the 80's. One of their first promoters was Richard Stallman, founder of the Software Foundation. Another important reference for this movement is Eric Raymond, founder of the Open Source. Their main texts *Homesteading the Noosphere* and *The Cathedral and the Bazaar* are really relevant.Without an anthopologic interest, and trying not to make an etnographic work, the author describes, with his limits, the relationship between workers that are in charge of the free software development. Relationships that are conceptualized as a *gift economy.*

Richard Stallman, founder in 1985 of the Free Software Foundation, created a production and exchange software community, based in cooperation. He wanted to remove the restrictions to the copy and modification of the computer programmes in a context where US was trying to change every open code into a propietary one. Stallman developed a new operative system, really different to Unix, called GNU (based on it, it means «GNU is not Unix »). The free software movement was born to get opossed to the commercialization of every kind of software.

The development of the free software movement made a radical progress when Linux Torvals, a Finnish student, wrote a kernel, or an operative system core, that was compatible with the Unix kernel, which comes from the propietary software. Torvalds filled an empty space in the GNU proyect, that started to be called GNU/ Linux. This new creation was not confined to expert knowledge. For this reason, every huge company, like IBM, HP, Sun Microsystems and Apple started to pay attention to it.

Within the software development field, we can find two different trends that are relatively opposite. In one side, the « free software » tendence and, in the other, the Open Source Iniciative, headed by Eric Raymond. The Open Source Iniciative, created by Eric Raymond, allows other people not only to modificate and redistribute the software. It allows to do these actions as if the software were a propietary one, something not allowed by the FSF. Open Source is more interested in the technological and economical profits that come from the open code.

Although Raymond's position is against the propietary software, his reasons are really different from Stallmans'reasons. Aparently, Raymond is not against informatic industry corporations. For him, the propietary software stops the technological innovation because it inhibits the possibility to create a best quality software. Stallman, in another possition, tries to defend users's freedom. He thinks they would have to know how the programmes use to work, although these programmes were not so efficient.

Because of this, Raymond makes a description about the labours that software developers use to do, and he proposes a theorization that justifies the creative software development, not seeing if the software was a free or a propietary one. To do that, he uses categories from the Economical Anthropology. Although he uses these concepts in a no specialist way (it's necessary to see that he is outside the academical world), these concepts allow him to caractherizise different practices within the movement.

Uses of cathegories from the economical anthropology in free software movement studies

To build a community

For Raymond, to build a free software project is to build a community. To this, he suggests making something that he calls "the bazaar's model". With this model, Raymond is making an implicit reference to Cleeford Geertz's work, in Marruecos (Geertz describes a "bazaar economy" where negotiations in a boisterous market of objects and practices substitute force imposition). This is what has been inspiring, and is inspiring now, software production.

Raymond proposes to make explicit the logical operation of software production, aparently anarchic, but not only to discover an unconscious dynamic. He wants, with an instrumental pretention, to improve the production. "… While creating programmes remains an essential lonely activity, the huge developments arise from the attention and thinking abilities of entire communities. Developer, that only uses his brain in a closed project is running behind the one that knows how to create an open and evolutionary context, where the check for errors and improvements are made by hundreds of people" (Raymond, 1997:25)

For some programmers, to produce software was the same as to build a building. It was necessary to build as if it were a cathedral. To guarantee that this project, that involves a lot of variables, doesn't collapse, they used to plan carefully each step. However, from his experience, Raymond notes that most succesful projects hasn't followed this path. These projects used to follow different rules, that were not clear but refered to a spirit of partnership and to some type of work that is not joint to a previews planning of all elements. The collaborative network became the general rule for all the projects, resulting in a more or less spontaneous community of developers.

Raymond proposes explicitly that the ex profeso creation of this community is now an essential condition to the "bazaar´s model" success: "It´s clear that you can´t start from nowhere in the bazaar's path. Within it, you can proove, look for mistakes, to tune up or fix something, but it´s really difficult to start a new project with the bazaar´s model. Linus didn't do it in this way. I didn't do it neither. Our born developer´s community needs something that is in race, right now, to run. When you start to build a communal building, you have to be able to present a plausible promise. The programme doesn't need to be particularlly good. It can be coarse, it can have a lot of mistakes, it can be incomplete or poorly documented. But something in which you can't fail is to convince to potential co- developers that this programme can improve into something more elegant in the future" (Raymond, 1997: 21)

Characteristics of the project leader, his qualifications, and the status of the code should be able to be evaluated by potential members of the community before they decide to participate in the project. But membership is not something that can be acquired directly or automatically. The hacker community recquires, Raymond says, that its potential members are able to understand the "reputation's game". As Raymond points: "You should, incidentally, claim that the "culture of gift" structure is the main mistery in itself. You are not considered a non cultural person (no one is gonna call you hacker) until you are able to

show some kind of level in understanding the reputation's game and its costumes, tabbos and usages. But this is trivial, every culture demands this understanding from the ones who want to join it. Adding to this, hacker culture doesn´t desire to keep its internal logic and its folk in secret (nobody told me anything when I reveal my practices)" (Raymond, 1999: 24). The kind of membership is only given by those already within the culture.

The hacker culture as the 'gift culture'

We should see Raymond's reasons to claim that we are looking to a 'gift culture'. The author says explicitly that 'to understand the role of reputation in the open code culture, it´s useful to see history within the athropology and the economy, and to review the differences between the "gift exchange" and the gift culture" (Raymond, 1999: 9).

For Raymond, the gift culture constitutes a model that apparently overcomes the hierarchical command and the exchange one. In the first model, the place of scarce goods is decided by a central authority, that is backed by the use of force. In our society, we use to see the second model, based in a culture of exchange. That's why "most people have implicit mental models for both of them, and how these models interact between them. The government, the militia and criminal organizations (for example) are hierarchical commanded models within the huge exchange economy that we call free market. Nevertheless, there is a third model, that is radically different from the ones below, and it's generally not recognized except for the anthropologists: the gift culture." (Raymond, 1999: 9).

These are the characteristics that Raymond gives to gift cultures: "Gift cultures are adaptations not to scarcity but to abundance. They arise in populations that do not have significant material scarcity problems with survival goods. We can observe gift cultures in action among aboriginal cultures living in ecozones with mild climates and abundant food. We can also observe them in certain strata of our own society, especially in show business and among the very wealthy" (Ray,ond, 1999: 9). For Raymond, in a different way from the primitive cultures where the gift is analized, in hacker culture people compite for prestige in a context of abundance, not scarcity. " Abundance makes difficult the command relationships to support and exchange relationships. In gift cultures, social status is not determined by the thing you control, but by the thing you give" (Raymond 1999: 10)

The "competition for prestige"

The main causes to look for prestige are related to certain general and particular characteristics of the hacker culture, that make this the only way to get 'status'. First, because for individuals it's a 'primary reward'. Second, because it's useful to "keep the attention and cooperation of other people. If someone is well known because of its generosity, intelligence, good treatment, leadership abilities, or other good qualities, it's easier to persuade other people that will benefit when they join you. Third, if your gift economy is in contact with

an exchange economy or with hierarchical commands, your reputation may pour down in them and win higher status within them too". (Raymond, 1999: 11)

Beyond these general reasons, peculiar conditions of hacker culture make that prestige would be more precious than a gift culture in the real world. The main specific condition is that the goods that are given are really complex. It's difficult to distinguish between a good and a bad present. Another peculiar condition is the relative purity of the open code culture. Most of the gift cultures are committeed, and there are a lot of reasons for that. It could be that they are related to exchange economies, like luxury goods business, or that they could be related to hierarchical commands, like family or clan organizations. There are no significant analogies of them in the open code culture; that´s why there's no way to win status if it's not by peers reputation. (Raymond, 1999: 11-12)

Property of the Free software project

The one who is recognized by the community with the right to ditribute the software and its versions, is the owner of the development. It´s the owner of a particular kind of goods, that are ideas, codes, algorithms, etc… things that we can't find phisically in nowhere, but in the noosphere.

Raymond says that "only when the modifications are placed to compite with the original in an open code community, this property turns into a point of dispute". Although the project property can be modified, the ways to do that are not direct. If project founder looses his interest or he doesn't do the technic support that is necessary, and other developers want to do that, the way in which the property is inheritated is discussed. (Raymond, 1999:5).

For Raymond, we must not confuse the noosphere with cyberspace. The noosphere is a kind of ideas reservoir, instead the cyberspace is a virtual space where these ideas are placed. The free software development requires "cultivating the noosphere". The discussion about the possibilities of software patenting, to grant intellectual property rights of informatic programmes, is a topic that goes through all these activities. Raymond defends the Open Source Inicitive, that tries to free contents within cyberspace from enclosures that are imposed by property rights. However, Raymond defends other kind of property in the noosphere field. Raymond defends the notion of property in the noosphere and links it with ethology. He says that "property is not only a social convention, but a mechanism critically important to avoid violence. To demand property (as to mark territory out) is a representative act, a a way of declaring which limits will be defend. When the community supports your property claim, you can minimize clashes and maximize corporative behavior. These things are considered true even when the property claim is more abstract than a fence or a dog barking, even when it is just to declare the project keeper's name, saved in the README archive. It's a territory abstraction (as other kinds of property) based in territorial instincts that has developed to asist and solve the conflict" (Raymond, 1999: 19)

Practices and taboos

To programme, to "sting a code" for programmers, constitutes the main programmers activity. The posibilities to change the code and distribute the programmes are the main reasons to be for the open source software advocates. Nevertheless, Raymond distinguishes between different tendencies within the hacker culture. "All members agree that open code (the software that is distributed free and can be easily develop and modificated to adapt into necessary changes) is good, and has a valuable significant and a collective effort. This agreement defines effectively the membrecy in the culture. However, the individual reasons and the reasons given by other subcultures about this believe, vary considerably." (Raymond, 1999:1)

Thus, Raymond establishes the distinction between "idealists" and "pragmatics" within the community. The first cathegory represents the defenders of an anti-commercial and anti- corporative view. The second is used for the less radical positions. "For pragmatics, the GPL is more important as a tool, not as an end in itself. Its main value is not as a weapon against the companies hoarding, but as a tool to encourage the software behavior, and to increase the development of the bazaar model within the community. The pragmatic person values to have good tools and toys, more than his antipathy by commercialism, and can use a high quality commercial software without having an ideological discomfort. At the same time, his open code experience taught stardards of technical quality that only a few software with closed code have" (Raymond, 1999:3). From development of Linux system, idealist positions were increasingly disadvantaged, according to Raymond. "The anti- trade purists were increasingly becoming a minority. It was not apparent how much the things changed until Netscape made the announcement, in February 1998, in which it said that navigator code from Navigator 5.0 would be distributed. This prompted to more interest in free software within the corporative world. The next called to hacker culture to exploit this unprecedented opportunity and to rename the "free software" as "open code" found an instant approval that surprised to anyone" (Raymond 1999:3).

According to the concepts pointed above, when Raymond analizes concrete practices of data processing workers, he points that they respond to a "promiscuous theory" of free software movement, that is opposed to the "puritan practice". The proof is the existance of certain taboos that indirectly confirm the operability of the Lockean theory of property, not only in material terms, but also in the virtual field. According to Raymond, practices of software creation of data processing workers refers to Locke's theory of property, and property habits are "means to maximize the reputation incentives; to guarantee that credits of our peers go where it's due and not go where it's not due. (Raymons, 1999: 12)

In theory, any person can hack or use any open code product. However, Raymond points that: "In practice, these divisions have never happened. Divisions in the main projects are odd., and they are always joint to public justifications. It's clear that, in cases like GNU Emacs/XEmacs division, or GCC/EGCS, or in the different BSD groups breakings, where the ones who broke felt that they were going against a rule strongly ingrained in the community." (Raymond, 1999:3)

The prooves are the three taboos of the open code culture that Raymond points:

0. "There's a huge social pressure against the project division. These breakings don't happened except a great need, with much public justifications and with other name. There's a huge risk to lose your reputation, "they can only control it, being in the both projects at the same time after the division (what is really confusing and not much practical)".

1. "Do not distribute changes in a project without moderator's cooperation, excepting in special cases such as the provission of trivial fixes in the software".

2. "Do not remove someone's name from the project credits if you don't have his explicity consent". If you do that, it's as if you were stealing this person's contribution in the project.

But besides the loss of prestige of the original development group, Raymond notes, there are other reasons. "First, hackers often explain their antipathy to divide projects. They can see that the division tends to break into two parts the community of co-developers, leaving these two 'son' groups with less brains to work"

Second, the not official patches "may complicate the bug tracking, and inflict work on maintainers who already have enough correcting their own mistakes" (Raymond, 1999: 12).

According to Raymond, although public hacker's declarations question the selfishness and make a cult of humility, they hide their real inclinations to the honor or individual prestige.

To recognize that, would corrupt the basis of the creative and cooperative spirit. Therefore, humility leads to greater productivity.

How gifts and presents are valued

Raymond points that there are rules to value informatic solutions within the community. The first is that the programme works as well as expected, although it may have mistakes or bugs. Second, an original work, that "increase the noosphere" is better than double an existing piece, unless it's a closed software. Third, the work that is circulating is better than if it is not. "The main distributions include, not only the huge Linux distributions, like Red Hat, Debian, Caldera, and S.u.S.E, but other distributions which are understood to have reputation by themselves to keep and certify quality (as BSD distributions or the source collection of Free Software Foundation)". Fourth, the work used by many people is better because it means that alternative solutions do not work as well as this one. Fifth, "the continuous devotion to hard and boring work (like debugging or writing documents) is more prestigious than making funny and easy things". And finally, "the extensions of not trivial functions are more prestigious than patches of low level and the debugging". (Raymond, 1999:18)

Property, project structure and conflict resolutions

To avoid conflicts, Raymond proposes a project structure based in the "benevolent dictator" model that, as we will see next, it's the prevailing model in the propietary software production too. "The not trivial easier case is when a project has multiple co- maintainers working under the command of a benevolent dictator that has the project. The habit makes this kinf of work easier for group projects; people work like this in huge projects like the kernel of Linux o Emacs, and it solves the problem of "who decides" in a way that is not worse than the other alternatives". (Raymond, 1999:21)

Dictator is expected to discuss his decisions with the co- developers: "While the benevolent dictator project harvest more participants, he tends to develop two lines of contributors; the ordinary ones, and the co- developers. A typical way to get into a co- developer is taking the responsibility of one important part of the project. Another way is to take the role of characterizing and fixing much bugs. In these ways, the co- developers are the contributors that make a continuous and substancial inversion of time in the project". (Raymond, 1999: 22)

Productivity in the SL project

After all the arguments, the true foundation of Raymond's theory appears: "The veredict of history seems to be that the free market capitalism is the best global way of cooperation for the economical efficience; maybe, in a similar way, the game of reputation within the gift culture is the best global way of cooperation to generate (and verify) a creative work with high quality" (Raymond, 1999:25). The reasons to support a non- propietary software are linked to productivity, not to challenge a system that is based on the privatization of common cooperation; "a group of open code development will be substancially more productive (specially in a long period of time, when the creativity becomes more critical as a productivity multiplier), than a group with similar size and knowledge of closed code programmers, (un) motivated for rewards." (Raymond, 1999:25)

According to Raymond, this is a result of the own programmers interests: "The 'utility function' that Linux hackers are maximizing is not economic in a classical sense, but something intangible like the selfishness satisfaction and the reputation among other hackers. (One could speak about their 'altruistic motivations', but would ignore the fact that altruism is for altruists a way to satisfy the ego)" (Raymond, 1997:26). Somehow, individualism is still there and is enhanced by the community. "I think that the future of free software will belong to people that know how to play the game of Linux, people that leave the cathedral model and take bazaar one in their hands. It doesn´t mean that individual vision and brilliance are not important anymore; in the opposite, I think that the vanguard of free software will be composed by people with individual vision and brilliance, that then would built positively enrich voluntary communities of interes" (Raymond, 1997:26)

Some considerations about Raymond's work

When we see Eric Raymond's work, we can understand that he uses categories from the Economical Anthropology, not only to see different practices, but to legitimate these practices that can be seen in the software production. That's the way in which a prominent member of the free software community perceives his work. Although that, Raymond's analysis is so closed to the liberal theories that defend the competitive markets (Adam Smith) and property (John Locke). That's why it's neccesary to pay attention to the *gift idea* because it can be exposed to different kinds of abuse. The Economical Anthropology was formed in opossition to these views that use to reduce the social complexity. Raynmond takes these categories and uses them in a different context. In this way, he can´t question the point of view of the involved people.

Something weird in his work is that the author refers to the neoclasical economic tradition to see the behaivour of the developers. The Neoclassic School defends the existence of rational and individual subjects that maximize functions, that point their objectives looking at utility and budget, without looking at the social or cultural context. This concept is not near from the real human practices thar are immersed in social conditions and meanings.But, in the same way that Polanyi criticized the "emergence of the liberal creed" and the notion of "autorregulated market" (looking at the classical economy of Smith and Ricardo), we have to refuse the use of the economical neoclassic conceptions to characticize the interaction between subjects outside and within the economic field.

Also, the author refers to liberal concepts to justify the appropiation of the technosphere goods. He uses the classical liberalism because Locke used to assert that work is the base of the property right. Although this, in Political Philosofy, is known that the contractualist tradition use to naturalize the property as an individual right. This argument is opposed to the idea that the knowledge comes from a common construction. The Marxist notion of *Generall Intellect* is much useful to analize these new dimensions of the appropiation (Benkler, Negri). We propose an approach to the economic field from the Critique of the Political Economy, close to new trends that come from Marxism, and that are useful to explain the work characteristics in XXI Century.

Although it's not a valid preceding, Raymond's work brings the possibility to get closer to different practices of the free software developers from the etnographic perspective. A perspective that can be useful to see this kind of relationships. These considerations about the etnographic pertinence can be transfer to the process of the propietary (non free) informatic work. Furthermore, we have to see if we can use some theories from another disciplines, like the Sociology of Work and the Critique of Political Economy, trying to make a richer dialogue between them.

Next, we'll try to show how important is to build a theory that keeps this kind of dialogue. This idea is enforced by Florence Weber, who thinks that the Economical Anthropology is not confined to primitive and traditional (not western) societies. This discipline is able to see the complexity that involves the informatic work, an essential part of the new technologies, within the last phase of the advanced capitalism (Weber, 2009: 29- 34)

The organization of the informatic work: Teams, projects and networks.

The informatic work consist in the software development, which means to construct different programmes as a final product. This product is different from other traditional commodities: the main difference is its intangible character. But this characteristic doesn´t stop the possibility to study these commodities in the same way as we use to see the traditional ones. We are able to study them looking at the different stages that constitute de "lifecycle", a very common informatic expression. These stages are: the requirements of the client, the design, the software architecture, the functional analysis, the parcial and global tests, the application and the maintenance (Castillo, 2009). In the other side, another principal characteristic is that within the process is common to work in teams, in relation with networks and looking at different projects. That's why it is important to collaborate with the network.

In the challenge of reaching specific objectives in a certain period of time, "project leaders" are necessary to coordinate the different resources. This "project leader" asumes the characteristic of the manager that doesn't give or wait for orders, they are "network's men"

About the horizontal work

The work in computer programming is shown as an example of a less hierarchical organization where knowledge flows. This is possible because of the substance of the final product. In general, writting programmes means to solve different "new" problems, to order values, to put them in efficient structures of memory, to give clear user's interfaces, to balance the charge of a set of processors, to exchange data between remote computers, to read and write different data format, and a lot of other labours:

For Martin, who programmes since he is 15 years old, to make a programme is "to sting a code", something that requires more skills each time. But although workers use more skills, they don`t have more rights to keep the product.

Martin: *Tools that we have as developers are different. Nowadays, they are quick, more complex, they allow you to atack… to make complex programs. A complex programme can't have a simple solution, you can´t solve it with actual tools. In the past we had other kind of programming, it was not based on objects, for example…*

Pablo*: Do you think that a developer has more requirements to programme, today?*

Martín*: Much more requirements…*

Pablo*: And, Is it always increasing? Is it possible to be different?*

Martín*: Technologies are evolving*

Pablo*: What happees with the training time? Is is easy to learn?*

Martín*: No, it isn't*

Pablo*: How can you make your activities consisten?t… Do you study more today than in the past?*

Martín*: Today, there is more information, more medias. If you want to be a developer, you can do it by your own. Even companies, to reduce the learning curve, use to select people that are capable to be trained in a basic level. They are prepared to learn company's working methodology, tools that are prepared to be used again. It's as if you have pieces, or black boxes or systems… we have to use these things that are made, they are guaranteed, they have a high quality, we know that they work.*

As in other works, Martin gives to his employers his "logical capabilities" but he doesn't think that the final product has to belong to him.

"You are working for your employer,… The code lines that you are writing don`t belong to you… you are only selling your knowledge that is reflected in the form of software….your logic capability… to generate code lines…. If you don't do it, someone is gonna do it for you…"

Pablo: But, why?… If each one do this work in a different way?

Martín*: Each one may do it in a different ways. That is why, one's solution may be brilliant and other's solution may be less brilliant.*

Pablo: *Don`t you fell concerned when you think that you have solved the problems and that the final product is not yours?*

Martin: *No, I don`t. Because a developer has to solve the employer's demands*

Challenges, rotation and leadership: the new types of exploitation (and the old ones)

Within the lower levels of the development process (the Junior level) is usual to rotate. This contrasts with the "implication" that is promoted by the project leaders.

For Daniel, chief of a little informatic development company with less than 10 employees, the division of labour means to recruit trained workers with more than technical knowledge. They must have other skills:

Daniel: *First of all you have de analysis, surveys, polls, tools to do a good analysis, of associated problems. Then there's the design, and this is a more expensive part, much more expensive, because you need more and better resources.*

Pablo*: Which are the resources?*

Daniel*: I need more programmers, good analysts that have good feeling with the client, they have to be able to deal with people, because they are with people that have problems.*

Although that, Daniel considers that rotation is unavoidable and that the leader defines the project success:

Daniel*: It´s related to the working teams, specially, the leaders*

Pablo*: Are you talking about the project leaders?*

Daniel*: Yes, I am, the project leader. Everybody looks at him as a leader, that's why it's necessary to work on that, to keep the working teams motivated.' "Leadership is generated in this area because it requires more knowledge*.

Pablo*: How is it possible that someone that is young, who is invested as a leader, is working with somenone older, someone thas has been working much more time?*

Daniel*: Generally in this item, leadership is generated by the ownership of knowledge. Someone that is distinguished as a leader use to have knowledge and authority, because he has more knowledge than others.*

For Daniel, to avoid the labour rotation, a good leader has to set challenges that workers have to be able to reach.

Pablo*: So, It's easier to change a developer than an analyst.*

Daniel*: Of course. When you have well organized your develop department, in a software develop company, developers use to rotate.*

Pablo*: Why do they rotate? Do they move to other jobs?*

Daniel*: There are a lot of reasons. Because there´s not motivation enough to work, there´s no motivation about the quality of the job they are doing. This can be a reason... they want to do something else, to use other languages, they want to learn other things. If they don't have interesting challenges, for example, if they don't have to do programmes for banks or multinational companies, jobs that they think are interesting... programmers go away.*

Daniel thinks that the leader determines the project success, but to achieve that, this leader has to select workers that accept the challenging labour and, at the same time, don´t distract in their work. For Carlos, a company manager that has less than 10 employees, workers salary is right, bur he would never do this kind of work.

Carlos: *That is something that you can learn. A guy is able to program in three years, and is earning a wage that any doctor will never earn. Doctors don't earn more than 3000 pesos each month, there could be some of them that earn more, but, on average, they earn less.*

Pablo: *But, Isn't it a tedious work for a guy?*

Carlos: *I will not do it… I'm not mad. But the analyst work, if someone gives me a really good wage, I would probably do it.*

Pablo: *Why?*

Carlos: *Because you are sitting in front of a computer, writing something everytime, everytime, I don't know. You are doing something with only one interaction, in fact, the less you relate with people during the working process, the best; because in that way you don't distract. If a man is thinking how to write a line, he can't talk with anybody, or relate with no one to do his job. Then, in a development unit, 'look! This is what happens', but once they give me my objectives, I have to write them, and I have to do it as far as possible and in a right way, to make a document…"*

For Carlos, to do a right job – "to sting code" as Martin says- requires low distraction, and a low interaction. Althought it's a dinamic area, this characteristics are similar to the industrial workers. Here, workers's rotation is not related to the absence of challenges. It´s related to the excess demand from workers and the relatively high salaries. Although that, Daniel recognizes that this rotation is a serious problem. Because it´s more difficult to find a programmer to replace the one that is gone, quickly. About this topic, he proposes different solutions that are really similar from other areas (technologically dinamic or not):

Pablo*: We are talking about a company that has only 20 employees, that is a pyme… I think that is a little company, with a little space… an office…*

Daniel*: That's true. When a resource go away there is a huge impact… I don´t know, two persons go away each six months*

Pablo*: How do you plan your team work when you know that is a little team with huge rotation?*

Daniel*: Each company has its strategies. It´s an important part of the strategic plan of each company. You have to bet to networks, to leaders of a development process, to a methodology that allows you to replace developers in an easy way. How do you do that? Making different tools, using development methods and a working team in which you can give less responsabilities to junior developers, you give them automated works. You take away the decision-making from them"*

Pablo: *"Don't you think that these methods can make the workers get worried? Don't they lose motivation?"*

Daniel*: "But, if they continue working, they will be promoted, and they will be leaders…"*

Working process had important changes with new informatic technologies, but explotation has not dissapear. It has accommodated to new circumstances. In Dyer- Whiteford's work, about workers in videogames industry, he points the main characteristics of new process in informatic works: "To conceive, write and programme virtual worlds, requires a synthesis of narrative, aesthetic and technologic capabilities. It requires to develop the different knowledges collected by the digital programmer, the graphic designer, the software tester, the set designer, the animator, and the sound and music engineer". And he adds, "In this 'inmaterial labour', that is not compatible with taylorist/ fordist technics of management, the game industry is the main place to test the team work, charismatic leadership, the ultraflexible time work, the open offices, soft hierarchies, *stock options*, a participative management of human resorces, and the *ethos* of 'work as a game'. This implies a soft leadership, a cool cooptation and a mistified explotation, with endless schedules, physical and mental exhaustion and chronic insecurity, without a trade union or worker protection (Dyer- Whiteford, 2004: 53).

Although our research has not finished, we have made some interviews and participant observation. Based on this information, we parcially conclude that the propietary logic of the software production can be explained by Raymond's nomination. The way in which the working teams for market production are

managed, shows similar characteristics with Raymond's description of the free software community. The struggle for prestige doesn't consolidate in higher salaries (that are relatively high), but in self overcoming and new challenges. The projects success depend on leadership skills, and its prestige (based on its knowledge). One thing that we want to analize is if Raymond's proposal doesn't legitimate the efficient production, instead of the free use of knowledge (efficiently or not)

The ethnographic view pertinence and the dialogue with other disciplines

Looking at previous information, we can see that ethnography is an appropiate method to see the practices within the free software movement and within the propietary software process of production. This ethnographic perspective is not based on technologies. It analizes the uses and perceptions that people construct about these technologies (Hine: 2008). The labours, relationships and meanings that are generated by people related to the social process of informatic production, can be implied or taken for granted, but they can be made explicit too. The assertions about these technologies (that are in medias, speciallized magazines, and academic papers) don´t analize the complex relationships and meanings that use to flow in the working process, where the job is more than a phisical conception and where there is an intangible product.

Thus, ethnography appears as a particularly useful method to analize the relationships between informatic workers, within the free software production and the propietary software field. In some way, in Raymond's texts we can see the use of Economical Anthropologic concepts in a not specialized way. With these concepts, he shows the informatic work and legitimate his considerations within the movement. In this paper, we tried to point that is really useful to merge in different trends, like the Economical Anthropology, the Sociology of Work, and the Critique of Political Economy, to reach a richer dialogue. In 1994, Arturo Escobar asked: "Could it be possible to see the multiplicity of practices with new technologies, in different social, etnic and geographic contexts, with an anthropolical view? In which way, these practices are related to broader social topics, like the control of workers, capital accumulation, lifestyle organization, and the globalization of cultural production?" (Escobar, 1994: 22). The categories that informatic workers create and put in practice consolidate a dialogue between the ethnography and studies of work and economy; because all of them considere workers as fundamental economic subjetcs (Weber, 2009: 12).

Bibliography

(2003) BENKLER, Yochai, "La Economía Política del procomún" en NOVATICA/UPGRADE, mayo/junio de 2003, nº 163, pp. 6-9, ATI, Madrid

(2009) CASTILLO, Juan José, "Las fábricas de software en España: organización y división del trabajo. El trabajo fluido en la sociedad de la información" en *Trabajo y Sociedad*, Nº 12, Vol XI, Otoño 2009, Santiago del Estero, Argentina

(2004) DYER- WHITEFORD, Nick, “Sobre la contestación al capitalismo cognitivo. Composición de clase de la industria de los videojuegos y de los juegos de ordenador” en Moulier Boutang, Yann, Corsanni, Antonella, y Lazzarato, Maurizio y otros (2004): *Capitalismo cognitivo, propiedad intelectual y creación colectiva*, Traficantes de sueños, Madrid.

(2009) DUFY, Caroline y WEBER, Florence, *Más allá de la Gran División. Sociología, economía y etnografía*., Ediciones Antropofagia, Buenos Aires.

(2005) [1994] ESCOBAR, Arturo, “Bienvenidos a Cyberia.Notas para una antropología de la Cibercultura.”, *Revista de Estudios Sociales* Nº 22, diciembre de 2005, Universidad de Los Andes, Colombia, 15-35.

(2001) HINE, Christine, *Etnografía virtual*, Editorial UOC, Barcelona.

(2007) JOLLIVET, Pascal: “Los rendimientos crecientes de adopción creativa ¿Hacia una competencia entre dos modelos tecnoinstitucionales en el sector software?” en (2007) RIVERA RIOS, Miguel y DABAT, Alejandro (coords.): *Cambio histórico mundial, crecimiento y desarrollo.*, Universidad Nacional Autónoma de México, México DF.

(1997) RAYMOND, Eric: “La catedral y el bazar”, Publicación electrónica disponible en http://biblioweb.sindominio.net/telematica/catedral.html.

(1999) RAYMOND, Eric: “Cultivando la noosfera” Publicacion electrónica disponible en http://www.geocities.com/jagem/noosfera.html

(2004) VERCELLI, Ariel: *La conquista del Ciberespacio: Creative Commons y el diseño de entornos digitales como nuevo arte regulativo en Internet.*, Publicacion electrónica disponible en http://www.arielvercelli.org/lcsdc.pdf, Buenos Aires.

(2006)YOGUEL, Gabriel, ERBES, Analía, y ROBERT, Verónica: “El sendero evolutivo potencialidades del sector de software en Argentina” en (2006) BORELLO, José, ROBERT, Verónica y YOGUEL, Gabriel (comps.): *La informática en Argentina. Desafíos a la especialización y a la competitividad*., UNGS - Ed. Prometeo, Buenos Aires.