How free software developers work

The mobilization of "distant communities"

Didier DEMAZIERE (CNRS, laboratoire Printemps, UVSQ)
François HORN (CLERSE, IFRESI)
Nicolas JULLIEN (MARSOUIN)

Free software are programs distributed with their source code (the text of the program written in a programming language that is comprehensible for humans) and with the authorization to modify and redistribute them freely, which differentiates them radically from private or "proprietary" software.

Their development is based on the participation of volunteers within a cooperative organization that relies a great deal on the organizational facilities provided by the Internet.

This configuration leads to questions on the characteristics of the collective action that enables the transition from individual voluntary commitments that are potentially volatile and unstable to the completion of a collective production that involves continuity and sustainability. The production of free software cannot be considered the contingent result of a spontaneous convergence of individual, independent commitments. It presupposes certain forms of motivation for the participants to work, who are in turn capable of ensuring a certain continuity in their commitments and of coordinating the organization of their contributions. Because even if a software program is a text, it is an "active" text that works insofar as it is made up of a list of instructions that are automatically executed by a machine, which requires an extremely strong coherency of the different parts of the text (Horn, 2004).

Empirical preliminary observations show that developers have a wide range of statuses (students, employees of research centers or private companies engaged in activities related to free software or not at all…) This infers heterogeneous links between the activity of developing free software and salaried work. The former can take place outside of working (salaried) hours, exclusively or not, but it can also take place during working (salaried) hours and thus can be, according to the case, hidden, tolerated, unofficial, official, required, recognized or valued. The development of free software takes place within plural legal and temporal systems.

These heterogeneous figures extend well beyond the scope of volunteer work and they indicate also another stake in this productive activity: the cooperation between contributors without which it would be impossible to develop a useable product. Yet, in general, these contributors are not enrolled in the same organization, are dispersed, have computer-mediated relationships via the Internet, and are not linked by the lines of an organization chart (Gensollen, 2004)

The absence of direct, codified and prescribed interaction between the producers is counterbalanced by sharing the sense of belonging to a specific group with a strong identity. At least this is how we can interpret the repeated references to "free communities" on the part of contributors. This indigenous terminology does not reveal its true meaning immediately, but it provides a clue to understanding the way the collective activity is carried out in the absence of organizational levers that usually make up the framework of work activities and the participants at work.

The work of free software developers is therefore both an individual activity carried out in extremely heterogeneous conditions and a collective action with original production methods. We propose to analyze this work starting with the paradoxical notion of a "distant community", that aims to illustrate the tension between, on the one hand, the strength of the sense of belonging to a specific world identifiable in the discourse of the participants and, on the other hand, the distances that separate the contributors in terms of relationships, status, and background. In doing this the aim is to produce a description, necessarily plural, of the different forms of "distant communities" that enables the production of goods in unique social and organizational conditions. More generally speaking, this notion points to methods of coordination that combine two forms of collective action that are usually contrary and antagonistic: a communitarian form based on the subjective feeling of belonging to the same community and a form of partnership based on the coordination of common interests and sharing of objectives (Tönnies, 1887, Weber, 1921).

At first, we will examine the ways the individual participants organize themselves in order to contribute to a project and we will focus on the forms of cooperation and coordination used to deal with the constraints of efficiency and quality associated with the distribution of a product. Secondly, we will look at the other side of the coin and examine the ways individual participants take action and we will underline the mechanisms of commitment and participation that account for their contribution to the production of free software. These two dimensions, that in our opinion are inseparable, are explored through a survey carried out with free software developers[1].

I.A collective project: organizing production from a distance

The production of free software is often carried out according to a plan where one person alone writes the entire program which is obviously limited in size. Even in this case, the updates, the correction of bugs, and the further developments can be socialized. And for more ambitious projects, which is the case of most well-known free software, the cooperation of several developers who write fragments of the program is required. Rules must be defined and decision-making bodies must be set up to organize the interfaces, distribute the work, combine the contributions, and edit the final product. Producers of private software distribute work according to organization charts and give assignments to a hierarchy that monitors the execution of tasks and coordinates the work of the developers. "Free" production has been analyzed as being founded on a "set of customs of cooperation that are the opposite of management by coercion" (Raymond 1998), or on strategies of free cooperation based on "give and take" (Printz 1998). These organizational forms that are barely hierarchical and hardly formalized are like the "model of a bazaar" compared to the "model of a cathedral" (Raymond, 1998) and reflect an emerging, more general, model referred to as "distributed knowledge" (Thévenot, 1997).

[1] In this respect our study is quite different from previous studies of the "motivations" of developers based on questionnaires (FLOSS, 2002). If we lose statistical width we gain in details of the process of participation of developers by linking them with the operating rules of the groups and projects within which they are applied.

Before examining the workings of real collectives oriented towards the production of specific software, we will describe several transversal properties that structure and organize the work of free software developers.

A.Linking isolated workers and coordinating individual production

Contributing to a free software program is essentially a highly individual activity, as Ernest[2], a Debian (Linux distribution) developer points out: "it remains a solitary job; Debian is 1,000 people working alone who make up a whole". In the same way, Linus Torvalds, the initiator of the Linux project, considers that "free software is made by craftsmen who are passionate about their art".

But in order for these different contributions to make up a software program, it is vital that collaboration be organized due to the properties of the product.

The aggregation of individual production in a collective significant and useful collective product is possible due to a series of organizing mechanisms of the production that, while they are different from institutional, coded or legal regulations, are efficient nonetheless.

The first mechanism is founded on a *rigorous modular structure* of the software that enables the creation of bits and pieces, and composition through assembling the fragments written independently by different people. This open and modular architecture is necessary because of the absence of a hierarchy with the power to channel and guide the work and contributions of the developers. Designed to facilitate cooperation between participants, it is also unanimously considered as a decisive factor for the quality of the software, but is rarely respected by commercial companies (Jullien, 2001). Moreover it allows for "the bulk of the architecture, implementation and creation phases of a software program to be carried out at the same time" (Brooks, 1996)

The second mechanism is founded on *two characteristics of the software*: the complexity of this technological object which means that a system composed of many developers working on the same program for a long period of time remains a phase of increasing efficiency and secondly the intangible character of software that allows it to circulate rapidly at practically no cost and which explains that all users can benefit from improvements without additional investments. The development of programs in the form of free software "engenders gigantic effects of learning by doing, i.e. taking full advantage of a fantastic *distributed intelligence*: millions of users that find problems and thousands of programmers who find how to get rid of them" (Foray, Zimmermann, 2001). In particular, this method of development is particularly efficient in eliminating errors, a task that constitutes a large part of the work involved in the creation of a software program. This is different from proprietary software that is more often than not revised by people very close to the authors and who make the same mistakes. A free software program can be examined by people who use a wide range of methods and tools which means that "each problem will be rapidly isolated and the solution will be obvious to someone" (Raymond, 1998).

The third mechanism consists in the *verification of individual production*. The setting up of one (or more) authoritative bodies that control and arbitrate between different contributions and select developments that are validated for integration in the software program is

2 The names refer to the free software developers we have interviewed. They have been modified to protect the privacy of the people we have met.

systematic in all the projects. The way they are established and the way they operate can differ, but their existence is proof of a formal and explicit organization. Thus Bernard describes a world that is “very, very structured. And then there is competition. Several people can propose different modules to solve a particular problem, and it is this group of decision-makers that for one or another component in the software is going to compare them and say: we’ll select this one and not the other. Therefore competition is open in intellectual terms, if I might say so, and after there is really a selection”. Individual production is not prescribed or ordered by a decision-making body but it is always evaluated and validated or rejected. We must however underline the fact that in this form of organization that is strongly horizontal even if it is not a totally flat network, the decision-making bodies have a unique technical legitimacy based on competence recognized by other developers and do not have any economic power over them. The absence of private appropriation of the software produced provides the possibility for a group of developers who are unhappy with the decisions made to develop an alternative project based on the existing software program (Himanen, 2001).

The fourth mechanism is *identification of the work* of each contributor: the lines of code are signed by their authors. The name of the developer is written near the parts of the source code on which he has worked and also in most free software programs there is a file entitled “credits” that lists the principal contributors to the software program and their participation. In a free software program the part that was done by each developer is publicly exposed which enables everyone to judge its quality. This point is very important because the qualities of a software program are not directly perceived through its use in that it is in fact a product in a system that interacts with other software programs and hardware components. In a free software program “the availability of the source code involves the programmer’s sense of pride because he knows that he is going to be judged by his peers. And for a computer programmer there are few personal satisfactions greater than having contributed to writing a program that is appreciated, used, taken up and improved over 10 years by thousands of programmers and millions of users because of its inherent qualities” (Di Cosmo, Nora, 1998)

These last two mechanisms allow for a fifth one: competition which influences the relation between contributors. Each developer can judge the quality of his work and his recognition: the selection of his proposal to contribute to a program, the choice of his suggestion for a correction, the integration of his program in a distribution, the number of times his software program is downloaded. Raymond (1998) insists on “the prospect of auto gratification by taking part in the action and being rewarded by constantly seeing (even on a daily basis) improvements of their work”. The visibility of contributors creates competition and “a situation where the only possible evaluation of success in this competition is the reputation that each person earns with his peers (…) The participants compete for prestige by contributing time, energy, and creativity” (Raymond, 2000). Taking into account the heterogeneity of the legal and temporal systems within which the developers evolve, it is not certain that this competition leads always to an intensification of commitment and to an increase in time and energy spent by each participant. But at least it contributes to regulating access to and the maintenance of this work and helps produce quality. In a way, the free software program model is organized according to the same principles as scientific research: free circulation of information that is criticized publicly, verification by peers, proposals for alternative solutions, and fierce competition between teams (Lang, 1999).

These regulatory mechanisms ensure that isolated or distant participants come together around collective projects. But they differ according to the project and thus configure differentiated organizational modes that we are now going to explore.

B.Different organizational systems, different social groups

Free software programs form a heterogeneous collection which has consequences on the methods used to produce them: the number and characteristics of the contributors, the organization of cooperation, the role and interest given to potential users. Thus, the general mechanisms identified earlier find special adaptations in each project. The way that tasks are distributed, the quality of programs is evaluated, errors are detected and corrected, and contributors are recruited corresponds each time to specific configurations. And each configuration can be considered as an attempt to build efficient cooperation and beyond that a minimal group solidarity between “distant” workers.

Certain characteristics reveal the organizational diversity of what we have decided to call “distant communities”: the size of the circle of principal contributors (which can moreover vary a great deal as the project evolves), but also the size of the other circles (secondary contributors who propose minor corrections, users who report errors); the characteristics of the initiators of the project, who can be individuals or public and private institutions of various sizes; the properties of the links that unite them, that can be limited to participation in the project or have been established before (network of alumni or colleagues in a particular field of study, a consortium of companies that have other objectives, etc.); the nature of the objectives and perspectives that reunite them and that can oscillate between multiple components that are not exhaustive (taking up a technical challenge, developing a market niche, defending certain values, etc.); the origins and the circumstances behind the project launch (improving particular functionalities, reviving a dormant project, planning ambitious objectives, etc.) We can only present here the elements of a few cases that are sufficient to suggest the range of organizational modes.

A frequently encountered case, in particular for small projects, is characterized by a hermetic and set hierarchy that is confined to the *monopolization of the decision-making process by one person*. Its workings are designed to delimit and maintain distance, not only in space but also socially, between the decision-maker and the contributors. This configuration is always founded on a singular story, that of an individual who writes a software program and proposes it to a file server. His product then is in contact with many users, who in certain cases can be very numerous, and some of whom do not fail to propose corrections, extensions or developments. But the initiator of the program tries to maintain the monopoly on the validation of further developments, and in some ways to relegate the other developers to secondary contributions (reporting errors, peripheral functions of the initial module).

As long as the contributions remain limited and occasional, the boundary remains clearly defined between occasional contributors and the initiator who is the guarantor of the product. The latter can thus reinforce his legitimacy and his recognition and maintain the monopoly, resulting from his initial personal initiative, concerning the free software program he created. The multiplication of the number of users and contributors, which is an indication of the growing success of the software, does not necessarily modify this organization because the initiator can form a small team by associating certain developers who are more regular or more significant contributors who will then control the contributions but also manage the contributors. An example that is close to this system can be found in the typographic composition software called Tex, created and controlled by Donald E. Knuth since 1978.

Another system corresponds to projects launched and piloted, at least during the initial stage, by a *group characterized by personal relations* between people that share common experiences or similar backgrounds. This social and/or spatial proximity of the initiators is often associated with a specific form of organization the basis of which is the image they have of themselves as IT professionals. This self-image is all the more solid in that the development activities are carried out in a professional environment. It then becomes highly effective and structuring in terms of the sense of belonging to a group and the definition of standards for the quality of the products. This preoccupation with the product introduces a pronounced differentiation between developers and users who are considered in some ways as the profane. This borderline is both distinct and permeable since the group of developers is not closed: outsiders who propose contributions that prove their technical competence can enter after cooptation, often confirmed by a vote by members of the group. The latter organize among themselves the distribution of the work, specialization in certain tasks, and definition of responsibilities for certain modules of the software program. One illustration of this is Apache software which is developed within the context of their professional activities by a group of computer programmers, systems administrators and software users of the Web server of the NCSA that was formed when the latter announced that it was dropping the product and stopping maintenance.

The efficiency of this type of organization has given rise to efforts to reproduce it with a core of initiators that is not made up of individuals but various institutions (companies, research centers…).These consortiums, the foundation of which can be encouraged by the government, group together partners who know each other through previous relations. The organization of developments is even more structured than in the case of a group of individuals. Here again the borderline is very clear between users and core developers, but the success of the first developments can lead to recruitment within the consortium of new partners which serves to amplify the project and reinforce its credibility. We can cite the example of the consortium ObjectWeb (a middleware platform), established by large French companies and research centers that has expanded recently to include American, German and Japanese companies.

A third form is organization around a *central institution* (a private or public company, a research lab), that initiates the project, allocates capital (in the form of salaried work), manages its development, and is in some ways the symbolic proprietor. However, the project does not remain confined within the framework of the institution and the circle of its employees as the principle of producing free software is to provide the source code of the program and therefore the possibility for any user to make his personal contribution. The choice of developing free software corresponds moreover to the desire of the institution that initiates the project to benefit from outside contributions. The institution that undertakes the project maintains, however, a central role in relation to the different circles of developers. It exerts direct and permanent control over the principal developers who are paid employees and linked by contract to the institution and organizes their activities.

As far as secondary contributions by users are concerned, they are examined and evaluated according to formal procedures. Generally, the participation of outside contributors takes place via websites and mailing lists devoted to the software program and can be structured by holding conferences. The evolutions of the software are thus all the more controlled in that outside contributors who are particularly productive and recognized by the decision-making body of core developers can eventually be recruited by the institution responsible for the software. Groups are therefore clearly segmented and the relations between core members are encysted within a professional relationship.

There are many examples of similar types of organization: research centers (the INRIA with the Scilab project), universities (University of Paris VII and Alliance software), companies (Zope software developed by an American company of the same name, CPS software developed by Nuxeo in France). Sometimes a company that edits a private software program decides to transform it into free software (Open CASCADE for Matra Datavision, Code_Aster for EDF).

Finally there is the case of *larger, more widespread and heterogeneous groups* that have modified their organizational rules as the group has grown in size to include members that are dispersed geographically and have no links due to social interaction. These groups of developers can include several hundred members which can create specific problems in regulating production and inevitably problems preserving the very identity of the group.

The initiators, who form the core, participate, in varying degrees, in the same social networks formed notably during school, but when the group expands this community based on common experience disappears and the social cohesion of the group is threatened. The growth of the group is both the result of the success of a product that interests many users, including developers, and the sign of a strategy of openness on the part of the founders. In this case, the longevity of the group and of the project is ensured by entry barriers in such a way that we can witness a paradox: the groups that seem the most open, i.e. the largest ones, are also the most exclusive i.e. the most selective. Recruitment is based on cooptation which ensures that all the members share the same technical competencies and values, as if this proximity of dispositions compensated for the distance between the positions occupied.

The fact remains that this improbable equation between the expansion of the group and selectivity for new members implies that the software produced is particularly attractive and creates more interest than usual. Moreover, these membership barriers help maintain less division of labor in the group and a sort of equality of situation or status so that any member can take charge of the organization of a given module.

The Debian project can be considered an example of this case (Auray, 2004, Conein, 2004). It has over one thousand members that all have the status of "developer-maintainer" with no hierarchy (a "project leader" elected once a year represents the project with outside partners but has no internal functions). Only individuals, excluding all institutions, can belong to Debian and membership applications are very numerous. Therefore a long and formal procedure has been set up that has several phases. Sponsorship by a member of the group, a technical aptitude test, and a test of the candidate's knowledge of Debian's philosophy guarantee that all the members share the same set of values concerning free software.

These examples show that the solutions adopted to organize distant production are highly diversified and reflect the constraints inherent in the projects developed, prolong the dynamics of the project launch and express the orientations of the initiators. The underlying issue of these various organizational modes is constant: creating a group made up of separate and distant individuals. To continue our exploration of this phenomenon it is necessary to understand what leads individuals to participate in this production.

II.The process of individual commitment

Most economic studies on the participation in the production of free software reckon that commitment is based on "classic" economic incentives, through financial valorization later on of the competencies of contributors to relatively successful free software programs: getting an interesting job, having privileged access to financial resources. This argument is based on the fact that a system which identifies precisely the contribution of each person to a free software program allows a developer to build a reputation that works as a powerful signal of competencies that are difficult to evaluate directly (Foray, Zimmermann, 2001, Lerner, Tirole, 2002). Our empirical investigations highlight processes of involvement that are more complex (cf. also Corsani, Lazzarato, 2004) and tend to confirm what Raymond wrote (2000): certainly " sometimes the reputation acquired (…) can spread in the real world and have significant financial repercussions [through] access to a more interesting job offer, to a consulting contract, or by attracting the interest of an editor" but "this type of side effect is rare and marginal (…) which is insufficient as a convincing explanation".

We have mostly met computer programmers for whom the commitment to free software had neutral, even negative consequences, from a material point of view. An extreme case is that of Ernest, a young computer programmer who left a consulting job paid 400 € a day to join a SSLL (*Société de Services en Logiciels Libres,* free software company) where he could spend all his time developing free software…for 1200 € a month. Of course it could be argued that his investment will be profitable later, but it seems that even when there are opportunities for financial rewards they are not systematically snatched up as we can see from the experience of Richard, manager of one of the first free software companies during the boom of the dotcom economy: "Imagine that in those days, like all the other free software companies, we didn't draw a salary at all or we allowed ourselves the minimum wage. We had companies like BNP and AXA come to us and say: you're a free software company. Would you like to…? So we said no. But we did hesitate a bit; there was a way for me, because I held 49% of the shares, to get several hundred thousand francs. And then the Americans VA Linux and Linux Care came to see us! And it was difficult to resist their siren's song. We held out only because we wanted to create a different kind of company".

Above all the validity of the hypothesis of motivation through financial incentives is founded on the premise of a contribution based on a calculated choice, anticipating the long term effects on a career. Yet, what our interviews show is that it is a more progressive commitment, sustained by a growing familiarity with programming activity and the "social world" of developers (Strauss, 1978) and accentuated by memorable experiences through which computer programmers build a sense of participation and interaction with other free software developers. If the individuals have their own, individualized production, this is a link in the chain of cooperation that, of course, organizes the specific technical know-how, but above all is personified in work habits, categories of perception, universes of discourse (Becker, 1988). Then, commitment to the development of free software is intelligible as a career choice.

A.The career of a free software developer

The interviews reveal several salient characteristics of a free software developer's work. It is organized in sequences that correspond to a succession of positions in the corresponding social world. Mobility from one position to another is the product of the encounter between personal motivations and integrating social environments. Career advancement corresponds to behavior that becomes stable and public and a reinforcement of the links of cooperation (Becker, 1963). Career progress does not only mean enrichment of technical competencies, but also the accumulation of social competencies involving ways of seeing and doing things, and codes that belong to each social world (Hughes, 1958). We have tried to identify the successive sequences that correspond to different modifications: in the behavior and activities of the individual, in the perspectives and meanings he attributes to his activity, and in the interactions and relations established with other developers.

Accessing the source code: increasing technical competence.

Development of free software concerns only those that are "passionate about IT" and who describe themselves as such i.e. people that not only have highly specialized and esoteric knowledge acquired through intensive use of IT tools and almost always a college degree in IT, but who also have a keen interest in programming. This frequently leads to the desire to access the source code of a given software program to correct the errors or make adaptations that were not planned for certain specific situations. A typical case is that of Stallman, the "inventor" of free software in response to a printer that kept jamming. He couldn't modify the software that was driving the printer in order to solve the problem.

A complementary source of motivation is the desire to understand how a software program works in order to learn programming. Thus Pascal explains: "the awareness of the importance of the phenomenon, of the importance of licenses, etc.., did not happen right away. That is to say, at first what interested me was only to have access, to be able to do things with it. I wasn't concerned at the time with cooperative development […] We had a systems programming course and I asked the teacher if by chance we could have the source codes of the Unix shell to see how it was made". On the same note Ernest told us: "when I started university, I said to myself: hey, at the university we're going to have to use Unix, so why not see for myself beforehand how it works. And then there was Linux, which is like Unix, which is free software that I should be able to install on my computer". Symmetrically, for many teachers learning computer programming requires being able to show how the programs are constructed.

Examining the source code of a software program seems normal to most computer programmers. But it is impossible in the case of private software. For this reason, computer programmers turn to free software in order to satisfy their needs or their curiosity. This initial phase of acculturation to free software is often encouraged by attending certain institutions, notably universities, which are historically favorable to free software. Even if this happens in an organized social context, it nevertheless is a response at this stage to a personal and often occasional need and it is disconnected from learning the significance associated with free software programs and from knowledge of how they are produced.

Producing a contribution and distributing it gradually

It remains that this acculturation takes place collectively, even if the geometry of the groups involved is limited to students enrolled in the same program of studies and their teachers. Some of the members of these groups, who often are only familiar with one particular free software program, are going to play a more active role. This process is in general very progressive. It usually starts by visiting the website of the software in order to follow its evolution and then is extended to participation in mailing lists which is often indispensable because of the initial difficulties involved in using free software. This participation, which consists at first of sending questions and can lead to proposals of answers to questions written by other users, enables the development of distant interactions outside the initial circle of colleagues and friends. The first contributions are often secondary: reporting bugs, translations and improving documentation…

These sporadic contributions and shared experiences enable the integration of a group and familiarization with its discourse which gives meaning to the actions carried out and can give rise to the desire for those who are competent to deepen their participation by proposing corrections and writing more important modules. The distribution of these first contributions is done gradually, by reaching larger and larger circles as the value of the production is recognized. The first recipients are the closest peers, then more distant colleagues but whom the contributor still knows personally, and then distant peers accessible through the website. This gradual distribution is a sort of initiation process combining a probationary period for the novice and validation of his production. Paul describes his experience of commitment to typography software: "Little by little, I became interested. There were things that I found, notably as a Frenchman, that didn't work the way I wanted them to, on a typographical level. So, I started to develop things and then to talk to colleagues I knew. It's not public; it's exchanging between people who know each other, let's say on an interpersonal level. And after, you submit that on public servers and it's recuperated by people that you don't necessarily know. But that's a second phase. That's not when you start. Well, obviously, the first stuff you do, it's like painters, you don't paint the Mona Lisa right away. So, you don't want to submit stuff that is going to be criticized by more competent people. I think that it's after a while that you say: Hey, that might be worth it. Finally, it's usually colleagues who say: You should submit that, really…"

In the first phases of a developer's career there is therefore a control mechanism through local networks of the quality of his production. When this is made public and available for all users, the person who produced it becomes a bona fide contributor because he has managed to participate in the reciprocal and social mechanisms of the products that are the basis of free software. He then assimilates the significance and the implications of his behavior. This evolution is enabled by the nature and organization of free software since the improvements that are proposed and accepted can benefit directly all users, the modified software can be used directly at no additional cost.

But beyond the technical conditions, it is truly a gradual and socially regulated process that allows an individual to attain the status of free software contributor. It then seems only natural to allow others to benefit from one's personal contribution when one has benefited from the work of other developers. Paul explains: "I started using it, I think like most other free software users. It's a thing that's available free…Plus its nice because it's not the fact that it's free but that it's open, that is to say, if their aren't exactly the functionalities you want in the software you can add them, modify them, so obviously it seems normal to share with the community of…If you have added something that can be useful for others, it seems normal that…You add it to the common pot, it's obvious".

Joining different groups and becoming a recognized professional

The final step of the process, followed by a minority of contributors, consists in becoming what could be called a free software "professional", i.e. someone who collaborates on free software projects during working hours, whether he is specifically in charge of this task, exclusively or not, or whether he manages to devote, more or less officially, a significant number of working hours to this activity. For this reason the free software "professionals" have a greater time commitment (in terms of length and stability) and make up the "core" of the communities that ensures the regulatory, organizational and structural functions described earlier.

This situation implies occupying a professional position compatible with a continuous and stable commitment to this collective activity. Working in these jobs can result in the gradual transformation of an existing job description enabling the developer to devote a growing share of his time to working on free software or the search for a new job that is in keeping with his participation in free software, sometimes after a period on substantial unemployment benefits. In the commercial world it can be the choice to work for a free software service provider, the creation of such a company, or more recently a job devoted entirely or partially to free software in a "traditional" IT company.

This professionalization is not only the institutionalization or the recognition of technical competencies. It corresponds to the acquisition of shared symbolic references and the adoption of specific values and beliefs that are the characteristics of this social world. This commitment to the development of free software is remunerated, but it is also often a commitment in favor of free software. There are thus strong beliefs that motivate a quasi-professional commitment in favor of free software, as expressed, for example, by Alain who, after having worked for a large IT company joined a free software firm and currently holds a job in a university where he devotes most of his time to free software: "let's say that for someone who has a technical profile, free software is great because if allows you to have control in society. You can have a political role; you can try to change the world by doing something in your field of competency. Belonging to a free software association, doing free software, is a concrete way of changing things and to say to yourself that you're not wasting your life, you know. So that's what makes me tick. I think it's the main motor for a lot of people".

On the other hand, the heterogeneity of the positions held by free software professionals suggests a differentiation in their backgrounds, their work and the significance that they attribute to their jobs. This is the point that we are going to examine now.

B.Contrasting reasons for commitment

Free software developers have above all been studied in terms of the diversity of their ideological motivations. Blondeau and Latrive (2000) reckon that they form an "improbable coalition" made up of "neoliberals, libertarians, Third-Worldists, and proto-Marxists". The main thing they have in common seems to be the will to defend the freedom of software users and to thus promote specific individual and collective uses: "the freedom to use the program whatever the usage; the freedom to study the functioning of the program and adapt it to your needs; the freedom to redistribute copies and therefore to help your neighbor; the freedom to improve the program and share the improvements with the public, so that the entire community benefits from them" (Stallman, 1998). These different ideological currents converge in the battle against monopolies, the biggest one being Microsoft. In France, this type of justification can be found in the existence of several associations that promote free software (APRIL, AFUL, FSF…), in the vivacity of exchanges (not only technical) that circulate on their mailing lists and between these associations as well as the many events that attract large audiences where both technical objects are presented (free software) and lively debates are held.

Thus the social world of free software is not uniform and career paths can be very different. We are going to explore this diversity of backgrounds and the meanings that are associated with them using material from four interviews with professionals selected for the contrasting points of view they present, the positions occupied, the activities carried out, the values championed, the beliefs defended and the network of membership.

A selfless activity akin to public research

Paul is a university mathematician. His first contact with free software resulted from his need for a typographic software program that could enable him to edit mathematical characters. However, as early as 1978, an American academic named Knuth had developed Tex over which he maintained complete control but around which numerous software programs were created, the most well-known being LaTex. LaTex is a free software program controlled by a small but changing team (mostly American in the beginning, its members are now exclusively European) and made up of academics and computer programmers working for scientific editors.

Paul, who was seduced by certain functionalities of this English language software, carried out some small developments to adapt it to the specificities of French typography and published them gradually. This is how he got in contact with the person in charge of the multilingual interface of LaTex with whom he collaborated closely. Progressively, Paul found himself taking care of gallicization modules and then developing other modules. This activity takes up more and more of his time in addition to involvement and responsibilities in Gutenberg, an association of French-speaking users of Tex.

His contribution to LaTex is closely linked to his job: "Was it during my working hours or my leisure time, it's impossible to say. But after all, even if it is during my working hours, if it's useful for the community it is no more useless than ideas I can have about math. I don't think that I have cheated on the state if I did it during work. And on the other hand, if I did it during my free time, since I had fun doing it, and in return I benefited from all the work the others have done on a volunteer basis, I think that I haven't been cheated". This interpenetration, even confusion, between work and free time has two different meanings that also converge. On the one hand, the software activity is an intellectual activity that should be part of "public domain", "exactly like research for the state that pays academics or others to develop free software". On the other hand his work as a LaTex developer provides him with satisfaction and quasi-professional recognition that he defines as more "rewarding" than research in mathematics: "In a way, I find it more rewarding to develop something that people use that to write a theorem that no one will use or maybe 30 years after I die. I enjoy it and its true that sometimes people tell me: Ah! You're the one that did that? I use it, I'm happy to see what you're like".

Paul therefore defends a model of development and publication of free software that he qualifies as "user-friendly" and efficient because it enables the production of better quality software. He also compares it very clearly against the market economy which according to him should not include the production and distribution of software because he sees the possibility of creating "different relationships between people. People come to see me and they buy nothing. I can help them and someone else will help me. You can call it a barter economy; you can say what you want, but it's still much friendlier". His opinions are shared by all the members of the community of LaTex users. Therefore he was violently opposed to one of the people in charge of Gutenberg that wanted to commercialize a gallicization extension for LaTex, an act which Paul considers as "betrayal of the spirit in which we all work". He personifies the categorical rejection of the software market and the refusal to use proprietary software and wryly refers to himself as "sectarian": "I don't want anything to do with it. Including the machines that I administer at the university. If you want to use

Windows, you can have somebody administer it, but not me. It's against my principles. I am for free software and therefore in my place there is free software. If you need something else, go see someone else. So, I do have a sectarian side, I admit it".

An alternative activity transposed, in the business world

Richard was passionate about computer programming at an early age. After university studies in IT and jobs as a traditional computer programmer in several large companies he created his own company in 1993 and developed "totally proprietary" software used to transfer information from Newton PDAs to company file servers. At the same time he followed the development of Linux (he was an Apple developer and "bought a PC just to see what it was").

The event that was going to make him switch definitively to free software was the decision by Apple to discontinue Newton in the beginning of 1998 which forced his company to shut down: "that day I said to myself: I'm never working on proprietary software again". He decided to redirect his business and create one of the first companies in France (and one of the only ones that is still independent) devoted to systems administration and specific developments based on free software.

Since Richard managed the company he had little time for development. However, he continued to develop in his free time a software program for electronic voting and collaborative publication for the internal needs of the company. The project that he started "for fun" grew bigger and he soon spent all his time on it, living on unemployment benefits after leaving the company after a drop in business. A first version as free software was published and Richard created a new company that commercializes services related to this software program.

Even if Richard works in the business world, he claims to be part of an alternative production model. Moreover, he freely evokes his past as a militant for the far left and considers free software as a "political stake": "It's still the first resource, the first product that is not on the way to being privatized but on the way to being socialized. We are privatizing water, soon air when it will be polluted. Well, here is a thing that's being created, and we say: look, this belongs to society". His political convictions are closely associated with his professional life, as if they were being carried out, transposed, and realized. Thus the two companies belong to the employees and the salaries are uniform. Furthermore, he has promoted the setting up of a network of companies related to free software that have identical values and that pool "all the information, whether in accounting, finance, economy, customers". This sharing of information claims to be a transposition of the organizational system of free software to the world of business. Because just as Richard is convinced that "free software sill supplant all the other software" because of the efficiency of its development system, he thinks that a network of companies owned by employees constitutes an economic model that will win out in the long term compared to traditional companies. He already points to as proof the greater resistance of this type of company to the recent crisis that rocked firms built around free software.

An innovative activity that corresponds to a commercial niche

Bernard has a different approach to free software. Even though he is also the founder of a company based on free software, he insists on the similarities with "traditional" companies. He was very concerned with questions of network infrastructure in his initial job as a computer programmer in a company and witnessed the development of the Internet "the very basis of which is the development of free software". He was convinced that with the success of the Internet free software would invade progressively the different "layers" of IT and "slowly permeate, through a viral process, the entire information system of companies and eject proprietary software from the market". He deduced an inevitable progression of the distribution of free software and saw in this activity the emergence of a sector of development worth promoting. But his hierarchy did not share his intuitions and he decided with some former acquaintances that were confronted with the same lack of understanding on the part of their employers to found in 1999 a company based on free software and which employs around 10 people today. The company's main business is the commercialization of system and network integration services by using numerous existing free software programs. The employees participate in communities created around these tools and submit "corrective patches" and software modules they have developed. The company has created a free software platform that enables all the applications of a company to communicate between each other no matter what their function or status (free software or not).

Bernard considers that "the strength of free software today" is that it constitutes a "new way of producing software": "companies that haven't understood that yet are going to be in deep trouble as time goes by, in that it's the same as sharing the cost of the R&D that there can be in the software. Before, you needed to put maybe twenty developers on line to obtain a soft. Today, you only need one person, or maybe two, knowing that you have the community working with you on the software". If groups that produce free software operate "informally", which is "not reassuring at all for rational minds that swear by the ISO label", they are, according to Bernard, "more innovative and more efficient" than traditional organizations ("today it takes an average of three days to correct a bug").

He is proud to belong to the "economic sphere" of free software that he compares against the "philosophical sphere" that he deems "sectarian". For him, debate about free software seems unproductive in relation to client companies ("free software is a problem between programmers") and his pragmatic attitude has led him to "insert free software in proprietary architectures" which shocked "free software purists" ("we have a pact with the devil"). A client needs to be "convinced that the free software presents a financial and functional interest, integrating a little bit of free software in his proprietary architecture and knowing how to show him that little by little we can insert a maximum number of free software programs in his network and information infrastructure".

A buoyant activity supported by intense militancy

The first contact Pascal had with the source code of a software program concerned a computer game and allowed him to understand how the game had been programmed. When he was a student at the ENS (*Ecole Normale Supérieure*) in France he learned about Minix, an operating system developed by an academic and the source code of which was public. Minix was rapidly replaced by Linux which interested him immediately and which made him aware of the strength of a "truly cooperative model" compared to development by an "isolated individual, however talented he may be and whatever his professional and intellectual competencies". His first contributions to free software happened within the framework of his first job as a researcher in mathematics: he proposed corrections and developed improvements for the use of a library program of mathematical algorithms.

In 2000 he created a company that currently employs 15 people. The company develops applications for clients (in particular administrations) by using a free applications server that was itself developed with a free programming language. Within this framework the employees propose corrections and contributions to the platform and the language on which the services are based and help to popularize them. Using developments carried out for clients, the company has created a "framework" that it distributes in the form of free software.

In addition to managing the company and organizing the community created around this software, Pascal has an important commitment and has had important responsibilities in one of the principal associations for the promotion of free software, of which he is a founding member. As he explains, "the aim at the start was to share something that interested me from a technical point of view, which I was even passionate about, and then progressively, it became a professional activity". This job of "popularizing free software, of preaching to managers and decision-makers, of helping counter attacks that can happen against free software" is complementary to his professional activity in his company that "is interesting because it encourages the development of free software on all levels". He claims to have a pragmatic approach to free software that after its initial successes will not become established on work stations without accepting to integrate proprietary software, going against those in favor of the exclusive use of free software. He criticizes developers of free software who are only preoccupied with the technical perfection of their creations without thinking about the needs of users. He is overjoyed by the progress in the way free software is made that combines "both a business and technical approach".

Finally the stories of Paul, Richard, Bernard and Pascal are unique: besides the different processes they use to invest in the development of free software, they attribute different meanings to this activity carried out in disparate biographical and institutional conditions. The sharing of a minimum base of competencies (particularly technical), of beliefs (in the efficiency of cooperative work) and belonging (to the same social worlds that they call "free") does not erase these differences.

Conclusion

"Free communities" constitute a paradoxical world because it is extremely open via the Internet and at the same time extremely selective and distinctive because of the competencies required of members. Our empirical results allow us to conclude that there is a great disparity of principles and rules of social organization of these groups on the one hand, and of spirits and significance of belonging on the other hand. However this diversity comprehends a common problem: how to produce a whole when we are separated; how do we create cohesion over such distances? The production of free software highlights specific work that can not be relegated to telecommuting or distance work on the part of the employees of the same organization, characterized by the cooperation between distant workers and free from the constraints imposed by an outside or collective authority constituted by being in a network.

We have tried to highlight the crucial stakes. The first concerns the creation of cooperation. We have identified the transversal mechanisms that ensure control over the work and the workers. Nevertheless we can find different interpretations according to the history of the projects and the groups that initiate and develop them. The second concerns that creation of commitments. We have identified general processes that shape the career of a free software developer. And this career follows different paths according to the individual's background and his social status. Thus the reduction of the distance between members takes on multiple social forms; and symmetrically belonging to a production group requires multiple social links.

The successive, but separate, analysis of these two dimensions enables us to note the tension between, on the one hand, the collaborative activity and the sense of belonging (to a group, a world, a community) that results from this participation and, on the other hand, the relational distance and the individualization of commitments that result from this isolation. The conclusions reached are temporary, but it appears in any case necessary to cross these two dimensions in order to obtain distinct figures of the paradox we have called a "distant community" and identify the segmentations of the free software world organized around individual forms of organization and mobilization.

Bibliography

Auray N., 2004, "La régulation de la connaissance : arbitrage sur la taille et gestion aux frontières dans la communauté Debian", *Revue d'économie politique,* Numéro "Marchés en ligne et communautés d'agents".

Becker H. S., 1963 (translated 1985), *Outsiders. Etude de sociologie de la déviance*, Paris, A-M. Métailié.

Becker H.S., 1982 (translated 1988), *Les mondes de l'art*, Paris, Flammarion.

Blondeau O., Latrive F. (editors), 2000, *Libres enfants du savoir numérique*, Paris, L'Eclat.

Brooks F. P., 1995 (translated 1996*), Le mythe du mois-homme: Essais sur le génie logiciel*, International Thomson Publishing.

Conein B., 2004, "Communautés épistémiques et réseaux cognitifs: coopération et cognition", *Revue d'économie politique,* Numéro "Marchés en ligne et communautés d'agents".

Corsani A., Lazzarato M., 2004, La fuite par la liberté dans l'invention du logiciel libre, *Journal des Anthropologues*, n° 96-97, 127-150.

Di Cosmo R., Nora D., 1998, *Le hold-up planétaire: La face cachée de Microsoft*, Paris, Calmann-Lévy.

Gensollen M., 2004, "Biens informationnels et communautés médiatées", *Revue d'Économie Politique*, Numéro "Marchés en ligne et communautés d'agents"

FLOSS, 2002, *Free/Libre and Open Source Software: Survey and Study, Final Report*, http://www.infonomics.nl/FLOSS/report

Foray D., Zimmermann J.-B., 2001, L'économie du logiciel libre: organisation coopérative et incitation à l'innovation, *Revue Economique*, 52, 77-93.

Himanen P., 2001, *L'éthique hacker*, Exils

Horn F., 2004, *L'économie du logiciel*, Paris, La Découverte.

Hughes E.C., 1958, *Men and their work*, Glencoe, Free Press.

Jullien N., 2001, *Impact du logiciel libre sur l'industrie informatique*, Thèse de doctorat en économie de l'Université de Bretagne Occidentale, 315 pages.

Lang B., 1999, "Ressources libres et indépendance technologique dans les secteurs de l'information", *Technique et science informatique*, 18, 8, 901-914.

Lerner J., Tirole J., 2002, "Some simple economics of open source", *Journal of Industrial Economics*, Vol. 52, 197-234.

Printz J., 1998, *Puissance et limites des systèmes informatisés*, Paris, Hermès.

Raymond E. S., 1998, *La cathédrale et le bazar*, traduction de Blondeel S., http://www.lifl.fr/~blondeel/traduc/Cathedral-bazaar/Main_file.html

Raymond E. S., 2000, "A la conquête de la noosphère", in Blondeau O., Latrive F. (editors), op. cit.

Strauss A., 1978, "A world social perspective", *in* Denzin N (ed.), *Studies in Symbolic Interaction*, volume 1, Greenwich, JAI Press.

Thevenot L. 1997, "Un gouvernement par les normes. Pratiques et politiques des formats d'information", in Conein B., Thevenot L. (editors), *Cognition et information en société*, *Raisons Pratiques*, 8, 205-242.

Tönnies F., 1887 (translated 1965), *Communauté et société*, Paris, Reitz.

Weber M., 1921 (translated 1971), *Economie et société*, Paris, Plon.