

# The Social Politics of Free/Libre and Open Source Software

## *An overview of social issues in the OpenSolaris project*

Thank you Barret for the introduction, thanks everyone for being here, to the Department of Sociology for hosting me in these months, to the Chair of the Department, prof. Harvey Krahn, who has welcomed me here and made possible this talk, and to Kelly Elkow for the logistic support, and to everyone I had the opportunity to meet and talk to in the last weeks. As you know, the title of this talk is “The Social Politics of Free/Libre and Open Source Software”, and my aim is to underline social issues connected with the techno-social phenomenon called FLOSS drawing on my two years of cyberethnographic research following the OpenSolaris project.

My first task is to give you a preliminary definition of what I mean with “Social”, “Politics”, “Social Politics”, and what is “Free/Libre and Open Source Software”. Later I will move to the empirical case in order to show you how the theoretical concerns emerged can give insights.

What is the social? Entering the debate about “the social” means entering a debate as old as sociology, as argued by Latour (2005), who firmly contrasts Emile Durkheim and Gabriel Tarde. Going back to the historical foundation of the discipline is an healthy exercise, because it allows me to make clear some theoretical concerns I have, as well as to trace connections between sociology as a discipline, the Science, Technology, and Society field, and the practice-based approach coming from Organization Studies which stays in the background of my view of Free/Libre and Open Source Software social world. As useful this travel to the origin of the discipline could be, as long it would become, so I should give you just few clues and concepts of the definition of the social I theoretically embrace and focusing more on the consequences for research. To be brief (and maybe too much schematic), I can summarize the difference between Durkheim and Tarde as a difference in the relationship between the *explanandum* and the *explananda*: if according to Durkheim the social is a kind of pervasive glue explaining other phenomena, the suicide for example; according to Tarde the social is what has to be explained by other glues, rooted in the monadic character of society. In this sense, society is what is assembled by connections of other kinds, and the task of sociologists is to

write accounts of the relationships between these connections, describing connectors, results of connections, and conditions for connections. Description becomes theory, and theory develops from a series of different connections. What does make connections possible and stable? This question arises automatically once I have pointed to the relationships between entities as constitutive of the social. The answer coming from thirty years of research in the STS field can be represented by the title of an article by Latour (1991): “Technology is society made durable”. Technology, artefacts, things, make the connections possible and stable. The task of sociologists become the explanation of the emergence of artefacts and things, in an effort to describe the processes of inscription of social ties in them and of de-scription of those same ties once they become technology-in-use (Suchman, 1994). Here the relational character of things has to be stressed again: technology-in-use doesn't mean only when the technology move from the social worlds of designers to that of users, but are the practices of use that, with their specific characters and aims, in whatever social world they take, place that allow to describe the inscription/de-scription related to the specific artefact.

These first remarks can help giving a double meaning to the word “politics” in the title I've chosen for this talk. On one side, politics becomes the inscription of views of the world in artefacts, the way through which a particular shape of society is translated into and by things; to quote Akrich (1992: 208), the act of inscription defines a framework of actions involving “actors with specific tastes, competences, motives, aspirations, political prejudices, and the rest, and assumes that morality, technology, science and economy will evolve in particular ways”. In this sense, the “social politics” means an analysis of the views of society that are translated and enacted by the ongoing technological phenomena. On the other side, inscriptions can be interpreted as characterized by different degrees of mobility, stability, and combinability (Latour, 1987: 236). That makes the analysis of the inscription of views of the world possible only if we can consider the mobility, stability and combinability of them, bringing sociologists to focus on the de-scription processes too, and on the ways through which the inscriptions read as de-scriptions of technology-in-use, allow the success in terms of stability of the views of the world inscribed. The dynamics of artefacts production and use become the centre of their politics, considered now as the ability to enroll allies making durable the society inscribed. To summarize, talking about social politics I refer to two interrelated aspects: on

one side, the inscription of views of the world in artefacts; on the other side, the ability to translate and enroll allies to make that inscription stable.

My ontological and epistemological standpoint is that views of the world emerge in practice. What is a practice? According to Gherardi (2005: 60 – 61), it involves: a qualitative and holistic aspect; a relationship with temporality; its being socially recognized; its being a mode of ordering the world. Let me focus on these points to better explain how do I approach the FLOSS phenomenon: with the stress on the qualitative and holistic aspect of the construction of a view of the world, I am brought to try to recognize the specific quality of the practice, in its relationship with established social worlds and discourses, as well as focusing on what allows to recognize the practice as a whole and to distinguish that as different from other practice; focusing on the temporal dimension means following how the development of the view of the world makes, unmakes and remakes society (Pickering, 1992); the social recognition pushes to focus on the institutional system that surround and is constituted by the practice; the attention to the way through which a practice orders the world is another time the stress on its social political consequences, as well as the focus on how practices stabilize themselves and what surrounds them. Talking about the construction of a view of the world as a practice, or as an ensemble of practices, makes possible to ask different questions, that are the leading ones while I face the FLOSS phenomenon: which are the social worlds, discourses, and elements at the origin of a view of the world? how does the view of the world change, and under which conditions? which is the institutional system which is enacted and allow the enactment of this practice? which are the ordering relationships connected to this view of the world?

It's with this lens that I approached Free/Libre and Open Source Software development, focusing on practices and their relationship with wider social worlds. What is FLOSS? To answer this question we have to step back to the beginning of the '80s, at the MIT, Laboratory of Artificial Intelligence. At that time, programmers where involved in the development of an operating system, Incompatible Time Sharing (ITS), and between them who has been called, in a romantic way, “The Last True Hacker” (Levy, 1984), Richard Maynard Stallman (RMS), was witnessing the effects of copyright laws on software development, on what could be referred to as “the hacker community”. It can be

useful remember the story Stallman told about how he became aware of what he conceptualized as menaces connected with the copyright system: in 1980, Xerox Corporation sent as a gift a printer to the MIT AI Lab, and that printer began to have problems. Referring to the usual practices between hackers, Stallman tried to find the source code of the printer device driver, in order to modify it, and solve the problems affecting the printer. Searching for the code, he found the original developer, and asked him for a copy of the driver. The answer was negative, due to the “non-disclosure agreement” signed between the developer and Xerox. This was, according to Stallman, the beginning: a consolidated practice of sharing of information, in the form of software source code, had been challenged, and modified by the copyright regime and the consequences on the relationships between programmers and employers. This episode lets the first of the social issues connected to FLOSS development emerge: how does the software industry relate to copyright regimes? how do the business models relate to them? how do these elements relate to the development practices? which kind of software development social world takes shape through the mediation of the copyright regime and business models? I will give some answer later, talking about the OpenSolaris case.

If this episode gives an image of how Stallman learned to connect the copyright regime to the changes in the hacker practices he was used to, his subsequent actions are interesting as well, and *de facto* constituted the beginning of what is now called “Free Software Movement”. Stallman started a problematization of copyright that he recognized as a way through which the “civic spirit” is undermined. The neighbours, hackers in that case, couldn't help each other. Another time, consolidated practices provided the resources for an action: Stallman was an operating system programmer, and he started writing software, in a project that he called GNU, acronym for “GNU is Not Unix” with the aim “to write a complete Unix-compatible software system called GNU (for Gnu's Not Unix), and give it away free to everyone who can use it.” (1983). Why do that? This was the Stallman's answer: “I consider that the golden rule requires that if I like a program I must share it with other people who like it. I cannot in good conscience sign a non disclosure agreement or a software license agreement. So that I can continue to use computers without violating my principles, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free.” (*ibidem*). Other actions followed that one, like the creation of a

Foundation, Free Software Foundation in 1985, and the construction of a software licence, the GNU General Public License (GPL, recently updated at version 3). I will talk later about the license, but primarily I want to go back to the pieces I've quoted, to remark two aspects: the choice of a Unix-compatible system and the ethical commitment, usually known as "the golden rule". The choice of Unix, which was not the operating system Stallman was working on, had two (later declared) aims: on one side, it's a modular operating system, with standard specifications, that allows developers working separately on different parts of the system, taking care only of the interface with other parts; on the other side, between hackers it was the most diffused operating system, enlarging the number of them who could be enrolled in the Stallman program (Unix was and is also diffused in computer science departments at universities). From the point of view of sociologists, this makes other two issues emerge: which is the role of the technical character of technology? does it act as a device of interest? how does the translation process (Callon, 1986) take place and how does it strengthen other programs? The "golden rule" introduces another aspect, the ethical dimension of practices. Stallman, grounding his action in his previous abilities and social world, gave an ethical meaning to his action and choices, entering an ethical discourse, and promoting a way of living ethically informed. Doing that, he was involved in what Foucault (HS) called "practices of freedom" (ECS). In Foucault terms, these practices have to do with the way the subject of reason defines itself as an ethical subject, and how it happens in the ordinary practices. In Foucault work, there is less attention to the process of liberation than to the practices of freedom, as a continuous effort to define themselves ethically, in a world of struggles about freedom and power, in the different discursive arenas where ethical issues emerge. That concept reshapes part of what I called the first social issue: how does the free software enterprise relate to an ethical conception of software development and use practices? how does this ethical commitment is framed? how does it change the software industry, in terms of the relationships between copyright, business models, and development practices? This opens up the space for a connection between the monadic and individual practical commitment and wider institutional and societal arenas of action.

As the time passed, in 1998, another group, whose spokesperson is Eric Raymond, developed and formalized a different problematization of the software industry and of the answers needed, bringing

to the creation of the term “open source” and of the Open Source Initiative. They shares more than one thing with the Free Software movement, in particular the attention to copyright, and the need to increase source code sharing. The reasons they give for that are different from the ones by free software advocates. Let's briefly analyse the commonalities, especially the copyright case. As I told you before, Stallman created a software license, the GNU General Public License, “hacking” the copyright system, referring to hacking as using the resources that are disposable, in that case the United States copyright regime, to change what was considered wrong in the same arena. The change, which has taken the name of *copyleft*, in a reversal of the copyright terminology, had the aim of warrant to the users of software (and not to the copyright owner) a certain amount and kind of rights, summarized by FSF as the four fundamental freedoms: the freedom to run the program, for any purpose (freedom 0); the freedom to study how the program works, and adapt it to your needs (freedom 1); the freedom to redistribute copies so you can help your neighbour (freedom 2); the freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). The open source group, still evaluating the free software effort, gave other reasons to embrace licences that could be considered free or open, relating more to the better technical results achievable with the enactment of a free/open source strategy than to the ethical and political commitment. This rhetorical and semiotic struggle, well defined by Berry (2004) as “the contestation of code”, developed in an interesting way, mainly to enroll corporations in the hackers world (the birth of “open source” is strictly related to the engagement by Netscape, Inc. with the hackers social world). As I will show during the discussion of the empirical case, these two groups, the free software one and the open source one, still enacting different “practices of freedom” in relation to the ethical accountability of their action, share a practical commitment to the development of the same technology and their contemporary presence is a fundamental element for the development of the reflection about the inscription of views of the world in the technology itself.

To summarize, I've underlined three main issues for sociologists of FLOSS: the relationships between software development practices, copyright regime, and business models; the role of the technical characteristics of technology; the necessity to give an account of the multiple translations

that are enacted in the practices of FLOSS. Now, moving to the empirical case of the OpenSolaris project, I will do two things: first, give an image of how those issues can be faced in this specific case; second, open up spaces for a reflection about new issues and the possibility of an ethnological approach, able to compare different projects.

The OpenSolaris project was opened to the public in June 2005, following the decision by Sun Microsystems, the company developing, maintaining, and commercializing the Solaris operating system, a Unix version, based on the Unix version by At&T and by University of California, Berkeley. At the core of OpenSolaris stands the development of a kernel, the software layer allowing communication between the hardware and the applications. Considering the kernel “market” in the FLOSS world, we have to consider the presence of three main projects: Linux, BSD (Berkeley Software Distribution), and OpenSolaris itself. Trying to understand how does the copyright regime relate to those three entities is a good exercise to focus on what I called the “first social issue”, or the relationship between copyright regime, development practices, and business models. Some scholars (Scweick, 2007) use to define FLOSS as a *commons*, an ensemble of common resources disposable for people who want to rely on them. Recent research (Hess, Ostrom, 2007) has questioned and in some way reversed the famous economic argument by Garret Hardin (1968) about the “tragedy of the commons”, arguing for a “comedy of the commons” (Rose, 1986) or a “cornucopia of the commons” (Bricklin, 2006). The main argument in favour of a rethinking of the tragedy of the commons is based on the difference between the kind of resources belonging to the commons: if Hardin was facing the management and use of resources, like grass in a pasture land, that are characterized by subtractability and exclusion, in the sense that the use by one participant subtracts resources from the commons and exclude others from accessing the same resource, the new scholars pointed that the new commons, in the form of knowledge or information, isn't subject to those two criteria, gaining value instead of loosing it with use. In that sense, FLOSS has been read as a huge commons of knowledge disposable in the form of source code, and one of the most used argument in favour of FLOSS is the absence of the obligation to “reinvent the wheel”, because re-use of good code is always possible. As Hess and Ostrom pointed, to make information, considered as a commons, subtractable and exclusive is an act of artificiality and production. Someone, or something

has to do that, but who or what? In the case of software, the artificial supporter of boundary creation between the resources on the commons is the intellectual property regime, mainly in the form of copyright regime (patent issues are relevant too, but not historically central, instead these issues are gaining momentum in the United States industry). Is the copyright regime making possible to consider Linux, BSD, and OpenSolaris as a unique commons of kernel source code? No, it isn't. And it isn't for reasons deeply connected with the relation between business models, history of FLOSS, and the choice of the software license.

In the explanation about why they are not a commons, let me forget about BSD, because it has a very permissive license, which allows the integration of BSD code into both Linux and OpenSolaris, but not vice versa. The boundary between Linux and OpenSolaris is traced by what is called “the viral clause”, which is part of the GNU GPL version 2, the license covering Linux, and of the Common Development and Distribution License, the license covering OpenSolaris. That clause obliges whoever makes modifications to the software covered by the license, and wants to redistribute the modified version (the clause doesn't apply if someone makes modifications for his own use), to redistribute it under the coverage of the same license. This legal mechanism denies *de facto* the possibility of mixing GPL and CDDL code, creating a boundary between Linux and OpenSolaris. The choice by Linus Torvalds to use GPL version 2 to cover his kernel is older than the one by Sun, in that case the incompatibility is an effect of Sun's choice, and inquiring this choice can highlight the relationships that I described as the “first social issue”, the ones between copyright, development practices, and business models.

To enter this topic, it's necessary to go back to the decision by Sun to open source Solaris, their operating system. According to Simon Phipps, Chief Open Source Officer at Sun, the decision was taken in 2001, as a consequence of the burst of the dot-com bubble, when Sun realized that “It became suddenly very obvious that lots of the people who didn't share Sun's values didn't belong here anymore; it also became very obvious that some of the approaches to software that Sun had been taking weren't actually in keeping with Sun's long-term values.” (Moody, 2007). Why the huge delay? A decision taken in 2001, becoming official at the end of 2004, with the opening to the public

in 2005? Why these four years? Is the same Phipps giving an answer: “software archaeology” (*ibidem*). With this locution, he refers to the fact that the code base of Solaris, also referred to as a “Wad of Stuff” (WoS), contained a lot of third party code, covered by different kinds of agreement between Sun and other companies, that required a close inspection and the ability to recognize, by lawyers and engineers, which part of the code can be released, being under the intellectual property of Sun, and which needed negotiations of new agreements with the copyright owners of the code. At the end, part of the code could be released as open source, and part couldn't. I want you to take particular attention to that, because here we can see one way through which the relationship between copyright, business models, and development practices, affect the FLOSS social world. In that case, previous existing development practices, including third party code without using what are called comments in software programming, lines of code not read by compiler programs but bringing information about the code, referring to the intellectual property status required to go back, and trying to make the different owners recognizable. At the same time, this was part of a business model involving proprietary software, so it was the absence of the need to show the code to external actors, and a strict intellectual property regime, that made efficient the choice of this kind of development practice. The open source choice changed things, pushing for a rethinking about these issue but, at the same time, the history of Solaris suggested to choose a license allowing Sun to still distribute the third party code which couldn't be released as open source mixed with the open source code (in what is now the proprietary version of Solaris, downloadable at no price, but not “free” as in the free software definition). As we have seen, the OpenSolaris license, the CDDL, include the “viral clause”, and this could make possible to think that the mixing between proprietary code and open source code can't be a choice. Here, the specificity of software license as artefacts helps us understand how this is possible, and why the Solaris Wad of Stuff resisted, for example, the possibility to choose the GNU GPL. Free and open source software license can be classified according to different criteria, the one relevant now is if a license is, like the GPL, “program based” or, like the CDDL, “file based”. The difference stands in the meaning given to the coverage of the license, standing the viral clause: program based licenses cover a program as a whole, so the adjunct of code to the program, in whatever form, is automatically covered by the same license; file based licenses

cover the single files of a program, and not the program as a whole, so if someone modifies something, but creating new files to do that, the choice of the license for the new files is not inscribed in the license itself, but it remains a choice of the writer of the new files (Rosen, xxxx). In the case of program based licenses, the possibility for Sun to mix the different kinds of code, is warranted simply moving the code they couldn't open source to specific files, and keep them proprietary. It's now clear one of the complex relationships between copyright, business models, and development practices, that can be visualized only with the image of a “field” in the sense of physics, related elements that act relationally on each other, modifying themselves reciprocally in a continuous tension. My analysis of the role of the license could go on for a while, but what I want to do is trying to explore another aspect of this relationship and then try to draw some partial conclusion about the “first social issue”, using the license case as an example. As I told you before, the license connect the present time to the past of the system, and of the software industry: in fact, Sun was founded in 1982, and at that time, except for the Stallman moral choice, the trend of the software industry was going toward a strong intellectual property system, mixing copyright, patents, and trade secrets (as far as I know, software is the unique kind of product that is covered contemporarily by these three different kinds of protection, at least in the United States, software patents in Europe are not recognized). Sun wasn't an exception, with the involvement in what are known now as the “Unix Wars”, or the constitution of two different alliances of Unix vendors, trying to establish two different standards, and with the proliferation of slightly different versions of Unix for any vendor. These wars were possible for two reasons: on one side, the original developer of Unix, AT&T, was constrained to license it's software to third party vendors by an anti-trust regulation in the United States, so developers had had for a long time (until mid '80s) the opportunity to look at the AT&T Unix code, and to develop their own version; on the other side, the Berkeley Software Distribution version of Unix was covered by a very permissive license, allowing vendors to add only small parts of code and make the software proprietary (as Sun did with SunOS, the father of Solaris). Without entering in a debate about the consequences of the Unix Wars for the software industry, what I want to point is the fact that, since the '80s, a bandwagon of elements, business models, intellectual property, and development practices were oriented toward what is called “proprietary software”, and we

have seen the consequences on license choice for Sun when deciding to free a product. But if one of the elements of the bandwagon has changed, the copyright in the case of Solaris, how are the other elements changing?are they resisting the changes?or allowing it?I previously want to focus on business models, then facing the development practices issue, for three reasons: first, Sun decided to open source Solaris when a business crisis happened, the burst of the dot-com bubble, so we can recognize in the business aspect one of the big tension creator in the Sun field of practices; second, the business model brings back again the history of the company, letting me underline again an important aspect, the boundaries history can create to the participation to the FLOSS social world; third, the business model enters the regime of accountability of corporations and let us understand which kind of problematizations and ethical consequences emerge for the company considered as a subject, giving us another image of how a corporation in the software industry can practice its own freedom.

Sun sells hardware. It has always been an hardware corporations, and their first product was what was called “workstation” (their acronym at the stock exchange market is SUNW, which stayed for “Stanford University Network Workstations”, and now stays for “Sun Worldwide”). Now, their main products are servers, based on their own processor, the SPARC processor (interestingly open sourced too, but talking about that will need another doctoral dissertation), and the services connected to the entrance of their machines in their customer organizations. This business model is almost old as the computer industry, with huge and expensive computers (I think everyone of you have seen at least once images of the first computers occupying an entire room), that made software an accessory for the same computers, needed to run them, but not generating revenues. The appearance of Unix, the first portable operating system, able to run on different machines, created a new possibility, to develop software untangled from hardware, and to sell it independently. This is the business model of the so called off-the-shelf software, generating revenues for companies like Microsoft (which almost half of the revenue is due to the Microsoft Office suite) or the anti-virus producers. As told, Sun business model is not an off-the-shelf one, but connected to provide hardware and services, goods that can provide enough revenue to make of the software a revenue – generator in an indirect way. That is what is happening in Sun, according to Jonathan Schwarz, the CEO of the

company: during the meeting for analysts and shareholders, he described the Sun business model as the one of the producers of radio antennas, which are more interested in people listening the radio than in making money with radio broadcasting. The way through which the business model is described is simple: increase the number of users of computers, between them increase the number of users of Sun software, the consequence will be a growth in opportunities for Sun itself, because of a growth of the need of servers and services, a growth of opportunities will bring to a growth of revenues, in a “volume drives value” logic. In this business model, software is the way through which market the Sun logo's and leadership, as recognized also by Goldman and Gabriel (2005), two Sun engineers who entered the debate about FLOSS as a business strategy, pointing also to the ability to capture unexpected innovations. From the point of view of my reading of the “first social issue”, this view of business model is important for a series of reason: first, this problematization of business and business model challenge what economists has identified as the typical economic dynamic in the proprietary software industry, known as “the winner takes all”, or the trend of an industry based on proprietary software (and standards) to become a monopoly, or an oligopoly; second, this problematization is possible due to the history of Sun, their commitment to selling hardware and software together, and to relate to a vision of computer as an infrastructure (the radio antennas); *third, this business model requires the opportunity to warranty about their own software to their customers, and the warranty is one of the services the Solaris version of the system carries with it, differently from the OpenSolaris one, another way of creating an alignment between copyright regime choice (the CDDL) and the business model.*

I've explored what I called the “first social issue”, now let me move briefly to the “second social issue”, the role of technical specificities. If, in the example above, this specificity appears clearly connected to the social politics of FLOSS, in the sense of the capacity of inscribing views of the world and to strengthen alliances around them, in the case of licenses; now I want to focus on another aspect: programming guidelines and code contribution. As argued, the modular structure of the Unix operating system, at the base of OpenSolaris, let different developers work separately on different part of the system. This could suggest that every developer can be free of choosing his own style of coding, in relation to the single part he is working on. But that is not the case. To see their

code included in the released version of the system, developers have to commit themselves to the respect of programming guidelines. Let's see now how this commitment is strengthened and enacted, and which are its relationships with the FLOSS social world, the history of the company and the organization of the project. First of all, the code contribution is filtered by organizational mechanisms: to submit code, non Sun developers have to send their patch, the modified piece of software, to Sun employees, creating a human filter, and Sun employees know the programming guidelines. These guidelines are the 1993 internal Sun programming guidelines, related to the use of the C language (the programming language that was born contemporary to Unix, with the involvement of one of the two original Unix developers), and one of the tasks for Sun employees is to refer to these guidelines while coding. In that case, the mediations between external developers and the code is another socio-technical assemblage, constituted by the downloadable source, the downloadable guidelines, the submission of the patch via email to a Sun employee, and the role of the Sun employee as a filter. In the struggles between different freedoms, this path enforces Sun's power, who has an employer – employee relationship with the human filter to the code submission. At the same time, this has been one of the more contested and discussed practices in the OpenSolaris community, which is now shifting to a non-human filter, what is called a Source Control Management system. SCMs work with different permissions granted to developers: if almost everyone has the opportunity to download the SCM version of the software, not considered enough stable to be released to the public, but integrating the last patches, and bug fixes, only few have a “writing permission” to include new patches and bug-fixes, writing permission granted via a path of inclusion filtered by humans. If it's still not the case in the OpenSolaris project, De Paoli and D'Andrea (2006), have shown how in the GRASS GIS community, the adherence to the programming guidelines is one of the things developers have to learn in order to receive the writing permission in the SCM. Coming back to the OpenSolaris community, two questions arise related to that: why not using immediately an open SCM (every software development company has its own internal SCM)? why referring to 1993 guidelines, and not creating new ones with the help of the external developers? These choices are clearly a way through which a view of the world about programming languages, part of the corporation practices, is translated in the FLOSS social world; but how does this translation relate to the general transla-

tion process connected to the establishment of OpenSolaris as a successful project in the FLOSS social world? It's not difficult to trace back the answers to both these questions to the established corporation practices: as we have seen OpenSolaris and the commercial Solaris system are two versions of the same software, with the commercial version tuned more before being released. Sun employees have been developing Solaris using their own SCM and their own guidelines for a long time before the opening of OpenSolaris, so it has been economically efficient to not choose an open SCM and new guidelines, in order to reduce the time needed for Sun employees to adapt to the new open source development practices. But the relationship between the community and the corporation practices is not an easy one, with the effort by the corporation to see the number of external developers grow, and external developers used to think at their own definition of what a "FLOSS community is". Let me read some passages for an email in the mailing list of the Community Advisory Board (CAB), the first governance body in the OpenSolaris project, posted by a CAB member, discussing the SCM too: "we discovered that the basic assumptions about collaboration were not shared by the most vocal folks in Sun Engineering. Instead, Sun's process will drive the basic goals and ideals of this new community, and we needed to wait until this process could be described to us outsiders because there really is no point in trying to impose self-governance. [...] We aren't creating an open community. Sun is creating a window into their own process so that code fixes can be directly contributed by customers to workers at Sun who will be tasked with shepherding their fixes." As we see, the *challenge* that the choices by Sun, like the use of the internal SCM, are a *threat* to the completion of the process of enrollment of new allies, undermining the same trial to make stable the OpenSolaris effort. On the other side, these kinds of remarks, connected with different translation processes enacted by the different participants, made possible that, during the first election of the new governance body, the OpenSolaris Governing Board (OGB), any candidate had in his agenda the implementation of an open SCM, and this process is now ongoing. What does it tell us? I think this tell us two things: on one side, entities, mainly people, are enrolled via a multiple set of devices, and ties, multiple "boundary objects" (Star, Griesemer, 1989), letting conflicting practices, and views of the world, staying together, creating a variable field of agreements and conflicts between the different entities; the pole of this entities, like that mail shows, or the debates about the license I

didn't talk about, show that the rhetoric is often moving between a group of supporters of more free and open choices (in the license debate often named “zealots”), and more “pragmatic” and slow approaches. That debate remembers, in its internal construction of “zealots” and “pragmatic” views, the one between Stallman and Raymond, the free software/open source one, the “contestation of code”. In any of these debate, we found two specific set of questions for sociologists, in relation to the social politics of FLOSS: first, if there are variable areas of conflicts, which are the ties holding together developers?how are they practically grounded, and technologically mediated?second, which are the consequences of this kind of debate for FLOSS development?are they “a cancer” as another developer define them in an email?if they are not a cancer, what are they?I'm not going to answer to the first set of questions, I hope the examples I gave you before can draw an image of the kind of legal, economical, and technical ties holding together developers. What I want to do is to try to give an answer to the second set of question. My answer is simple: these debates are the essence of FLOSS. Why do I say that?I've tried to show how there are a lot of different ties holding together developers, with conflicts in specific areas, with variable alliances and view, rooted in different practices and aims, so these debates, when involving a lot of people (and I've found in more than two years less then 10 of this kind of debate, lasting few days, with a huge frequency of mailing-list messages), are difficultly disrupting the other kind of ties, whatever is the strength of the arguments. On the other side, these debates are the occasions for a problematization of current community practices, for a reflection about the inscriptions of views of the worlds in the different artefacts the community is producing or using, mobilizing forms of knowledge that often refer to other social worlds, the legal system, political theory, etc..., that increase the awareness of technology connected topics between the same developers. At the same time, the construction of the two different groups, enacted differently in any occasion, maintains a tension between the entities in the field and pushes the corporation to become accountable in relation to the various issues (Phipps's essays about licensing and the CDDL, Schwartz often entering debates about views of society, the move to an open SCM, are all examples of the different answer the corporation is giving to the different issues). This is also related to what I told in the introductory part about practices of freedom, and a possible case of re-framing this concept in relation to FLOSS: Sun is in its own practice of freedom in the Fou-

cault sense, it's becoming an ethical subject, but this ethical stance is relationally negotiated with the other participants to the FLOSS social world and the OpenSolaris community; doing that as a strategy to market the project, trying to enrol more developers as possible; the presence of different groups of freedom/openness “zealots” push them to become ethically accountable, and to engage in continuous changes in their translation strategy; these mutual engagement, with the presence of a field of translations and strategic moves, as well as the presence of other kind of ties, make the OpenSolaris community “a field of practices of freedom”, a space in which the different ways of enacting their own freedom foster debates about freedom itself, constructing a space of continuous actions and opening of new fields of debate, and freedom for other participants. The boundaries around and between the community are continuously re-negotiated and modified.

*These words, let me conclude facing the third social issue, the question about how the translation process enacted open sourcing a program take place.* I've used this term, without clarifying what I mean with it, for the whole talk, now I need to become clear and come back to theory for this kind of issue, to let other issues emerge, and give a first sketch of an ethnological approach to FLOSS, that is now in my agenda. As translation process, I refer to the argument by Michel Callon (1986) and other actor – network theorists (Latour, 1987; Law, 1986). This process has to do with the landscaping of the world connected with techno-scientific enterprises, in other way is a process to affirm the truthfulness of a techno-scientific view and its reality. In actor – network terms, it's a way through which a network is established, and one or more entities become spokespersons able to mobilize the network to make the techno-scientific reality stable. Briefly, this process involves four analytically different, but practically contemporary and mixed steps: problematization, interessement, enrollment, and mobilization. The problematization is the formulation of a problem, remembering the Foucault theory about knowledge; the interessement is the construction of devices able to stand between entities, and to tie them to the entity taken as a protagonist of the process, separating them from competing programs; the enrollment is the distribution of roles and delegations between the network; the mobilization takes place when the spokesperson is able to speak for the other entities in the network. Let me come back to the OpenSolaris case, and try to give a picture of how this happens, and how it's opening spaces for the action of FLOSS social world outside its own

(undefined) boundaries. As I've shown, the problem for Sun was a business problem, a problem of company identity inside the computer industry, and FLOSS social world has been seen as a useful entity to be enrolled in the Sun strategy. This pushed on one side Sun to elaborate a problematization able to cross the edge between the investors and share holders social world and the FLOSS one; this has been done through the construction of a business model, the “radio antennas”, that allowed Sun to be accountable in front of share holders and, at the same time, to embrace the copyright regime typical of FLOSS; the reading of the FLOSS social world by Sun, let other competing and parallel problematizations emerge, technological ones, and societal ones (I've not talked about them, if you have question we can discuss them later), as well as the construction of different and numerous devices of interessement, from the CDDL license, to the website, passing through the mailing lists and the system itself; the multiplicity of the devices of interessement, as well as their different degree of robustness, make them boundary objects, able to interest some entities, and exclude others (like the Linux kernel code base, for example); the enrollment has been set up in multiple and variable ways and through multiple and variable set of roles; the mobilization is always incomplete and unstable, but this instability, like the contrast between different kind of “zealots” and “pragmatists”, is at the core of the formulation of new problematizations and new ways of interesting/enrolling/mobilizing allies. Taking this preliminary conclusions, an ethnological approach to FLOSS has to consider: the different forms of problematizations connected with the beginning of a project; the strategies of interessement, enrollment, and mobilization, as well as how this strategies re-frame the problems at the base of the different translations processes; the practices standing behind, and related to, every kind of translation taking place; the multiplicity of the translations enacted, and the discursive resources mobilized, with particular attention to the different social worlds referred to. In one sentence, an ethnological approach to FLOSS has to give an account of the multiple, unstable, unfinished practices of techno-social reality construction taking place.

This attention, as some of my words could have make you think, lets other questions arise for sociologists, as complex as FLOSS is, like the conceptualization of FLOSS as a social movement, as a business enterprise, its ability to relocate power in the software industry, its feasibility as a model for other social worlds, its relationship with political theory, etc... certainly topics that need to be

part of a research agenda about FLOSS, and that I would like to focus on in the next years. Thank you.