WHY FREE SOFTWARE IS NOT THE ANTONYM OF COMMERCIAL SOFTWARE: TWO CASE STUDIES

ABSTRACT

Academic literature often uses the terminology "commercial software" as an antonym of Free/Libre Open Source Software (FLOSS). In this paper we challenge this opposition showing that in FLOSS stakeholders discursive practices there is a mix of different gradients of FLOSS and commercial. In particular, we propose examples taken from two case studies: the Geographical Information System known as GRASS and the Operating System known as OpenSolaris. GRASS is a system covered by the GNU/GPL software license and developed by a community of volunteers. OpenSolaris was instead backed by one of the major IT world player Sun Microsystem. Using a practice-based and a socio-technical framework the paper illustrates how FLOSS developers, both volunteers and corporate employees, sees the terminology"commercial software" as a constituent part of their activities and of what they produce. In this way we aim to provoke a reflection on the use of words in FLOSS academic literature, deepening the understanding of FLOSS discursive practices.

Contents

Introduction: Commercial or Proprietary Software?

In defining what is Free/Libre Open Source Software (hereafter FLOSS), the academic literature often opposed FLOSS to the terminology "*commercial software*". To a certain degree this opposition also takes for granted that "*commercial software*" is a synonym of "*proprietary software*". In this paper we criticize the opposition between the terms FLOSS and commercial software and shed new light on the use of the terminology *commercial software* as being instead a defining component of FLOSS[i] rhetorical discourses. In particular, by the means of two empirical-qualitative case studies we illustrate the process of co-construction between the commercial meaning of FLOSS and the social and discursive practices of software development.

FLOSS is an approach to software licensing that provides the users with the ability to change and share the software (these abilities are called freedoms of software by the Free Software movement – see Free Software Foundation, 2004). FLOSS licenses allow the user to use, modify and distribute both the source code and the object code of software. FLOSS is also an innovative software development methodology in which a flat and distributed organizational model (typically known as Open Source, Bazaar or Linux Development Model) is opposed to a hierarchical and centralized development model (defined by Raymond, 1999b, as Cathedral model). Notable examples of FLOSS projects are the operating system GNU/Linux and the Web Browser Firefox.

A definition of FLOSS should consider the opposition to the so-called *proprietary or closed source software model*. The latter is a software development model in which the producer bases the business on selling copies of the software in exchange of money[ii] (sells a license that allows the user the execution of the object code). Further, proprietary software is developed in full secret only by the producer's employees, with an exclusive control on the software source code which remains closed for the users. Notable examples of proprietary software are the operating system Microsoft Windows® or the Adobe Acrobat Reader®[iii].

In many – and often very influential – literature contributions, FLOSS is defined as an antonym to the terminology *commercial software* and hence by extension commercial software seems to be a synonym of *proprietary software.* In our view this opposition between FLOSS and commercial software (and the subsequent identity between commercial software and proprietary software) creates in fact a discrepancy between the existing academic literature discourse and the ways FLOSS developers use the word commercial to explicitly refer to their activities. In other words, despite the academic literature often portrayed FLOSS as an antonym to commercial software, FLOSS developers seem to define what they do commercial software. In *Table 1* below we summarize a number of literature cases in which this opposition between FLOSS and commercial software is used and these include also most of the papers published in the recent Special Issue on FLOSS (December 2010) of the Journal for the Association of Information Systems[iv], a clear sign of how actual is the issue. Moreover *Table 1* shows that this opposition between FLOSS and *commercial software* is grounded in different aspects of the FLOSS debate, including: the problem of different licensing schemes, the different organizations of software development, the degrees of openness of the source code, the efficiency of FLOSS software compared to proprietary software and finally even from a purely historical viewpoint.

Our critique to the literature and the use of the terminology commercial software an antonym to FLOSS is based on a simple consideration: the literature does not take in account the meaning and the use of the terminology commercial software for FLOSS stakeholders themselves, and in particular for developers. Therefore this misunderstanding is the outcome of a lack of practice-based research on FLOSS (Lin, 2005), and in particular empirical research that takes seriously in account the FLOSS stakeholders point of view. On this specific aspect we challenge the mainstream literature by arguing that FLOSS stakeholders use the terminology commercial software to define FLOSS as part of well defined and concrete strategies of both software development and community building. In this sense, we agree with Wheleer (2006) for which the opposition between FLOSS and *commercial software* is not only imprecise, but also mistaken for several reasons:

> (1) the rise in commercial development and support for FLOSS, (2) most FLOSS projects' goal to incorporate improvements (which are actually a form of financial gain), (3) official definitions of "commercial item" that include FLOSS, and (4) FLOSS licenses and projects that clearly approve of commercial support. Terms like "proprietary software" or "closed source" are plausible antonyms of FLOSS, but "commercial" is absurd as an antonym.

In brief, with this paper we contribute to augmenting Wheeler considerations with a serious empirical and practice-based investigation that highlights how the terminology commercial software is used by FLOSS stakeholders to make a strategic order and sense of the social worlds they inhabit. We propose examples taken from two different case studies, that provide us with a good degree of variety: (1) the development of a Geographical Information System known as GRASS, a system covered by the GNU/General Public License and developed by a community of volunteers and (2) the Operating System known as OpenSolaris backed by Sun Microsystem, that was one of the major Information Technologies world players.

The paper is organized as follows: we initially describe our approach, including the theory framework and methodology (sections 1 and 2); then we present the empirical cases of GRASS and OpenSolaris (sections 3 and 4); then we present a final discussion of findings and a conclusion.

Table 1: Some relevant examples of how FLOSS and Commercial Software are used as antonym in academic literature [*italic emphasis added*].

Historical viewpoint:
A good way to get a first grasp of open source software is to observe how, throughout its history, *it has differed from commercial software*. (Von Hippel and Von Krogh, 2003a, Online version)

I briefly discuss the case of *Linux entering the markets for server operating systems previously dominated by commercial software enterprises*. (Bitzer, 2004, Online Version)

Over the past decade *Free/Libre Open Source Software (FLOSS) has become a viable alternative to proprietary commercial computer programs*. Fueled by the rise of Linux and open standards such as HTML and Java in the 1990s, the concept of Free/Libre Open Source Software development permeated the Information Technology world during the early and mid-2000s. (Chengalur-Smith *et al*, 2010)

Licenses viewpoint:

Open Source Software is given away for free by the developers who write it, both in the sense that it is provided at a nominal charge and that *it is licensed to users without the legal restrictions typical of commercial software.* (Healy and Schussman, 2003, p. 2)

However, the fact that open source software is freely accessible to all has created *some typical open source software development practices that differ greatly from commercial software development models*—and that look very much like the "hacker culture" behaviors described earlier (Von Hippel and Von Krogh, 2003b, p. 211)

Transactions among agents in an open source environment are regulated by a variety of licence agreements which, in different ways and degrees, *protect the openness of the source code and prevent the commercialization of cooperatively developed software*. (Lanzara and Morner, 2005, p. 86)

Development Model viewpoint:

On the other hand, *a major difference from commercial software development is that, in open source projects, the requirements are not fixed over the life time of the software*. According to the requests of programmers and especially users, new functionality is added. This violates the assumptions of most traditional models for software development effort estimation. (Koch S. and Schneider, 2002, Online version)

If we look at *the amount of code produced by the top Apache developers versus the top developers in the* *commercial projects*, the Apache core developers appear to be very productive, given that Apache is a voluntary, part-time activity and the relatively “lean” code of Apache. (Mockus *et al*, 2002, p. 324)

[…]

In the “free” world of OSS, patches can be made available to all customers nearly as soon as they are made. *In commercial developments, by contrast, patches are generally bundled into new releases*, and made available according to some predetermined schedule. (Mockus *et al*, 2002, p. 330)

Our study examines distributed development in the context of one commercial entity, which differs greatly from both open source projects and outsourcing relationships (Bird *et al*, 2009).

Recent decades have witnessed the success of Open Source Software (OSS) development [...]. Major companies such as IBM, Oracle, and HP, as well as large venture capitalists, are investing generously in the communities that develop OSS [...]. In the meantime, *researchers and practitioners have begun asking questions about how and why this practice can succeed without the same control mechanisms as commercially-produced software* (Ke and Zhang, 2010, p. 785).

Source Code Access viewpoint:

From an economic point of view Open Source software can be analysed as a *process innovation*: a new and revolutionary process of producing software based on *unconstrained access to source code as opposed to the traditional closed and property-based approach of the commercial world*. (Bonaccorsi and Rossi, 2003, Online Version)

Later, *when commercial software development increased and often only the software vendor had access to the source code of a program, OSS became an attractive alternative* since it enabled the users to adapt and improve the software according to their personal needs. (Hertel, et al., 2003, Online Version)

Most commercial software is released in machine language or what are called "binaries" — a long string of ones and zeros that a computer can read and execute (Weber, 2004, p. 4).

OSS seems to be a unique opportunity to enhance our knowledge about the role of individuals in successful reuse-based innovation and software reuse, in particular, for two reasons. First, *contrary to commercial software developers*, who are often restricted to the limited amount of code available in their firms' reuse repositories, *OSS developers have broad options to reuse existing code* if they wish due to the abundance of OSS code available under licenses that generally permit reuse in other OSS projects. (Sojer and Henkel, 2010, p. 870)

Efficiency view point:

On first examination, open source software seems paradoxical. Open source software is a public good provided by volunteers—the "source code" used to generate the programs is freely available, hence "open source." Networks of thousands of volunteers have developed widely used products such as the GNU/Linux operating system and the Apache web server. Moreover, *these are highly complex products and they are, arguably, of better quality than competing commercial products*, suggesting that open source provision may be highly efficient. (Bessen, 2005, p. 1)

Perhaps in the end the open-source culture will triumph not because cooperation is morally right or software "hoarding" is morally wrong (assuming you believe the latter, which neither Linus nor I do), but simply because *the commercial world cannot win an evolutionary arms race with open-source communities* that can put orders of magnitude more skilled time into a problem. (Raymond, 1999b)

For well over two decades, people have debated the merits of developing and distributing software under what has become known as the "open-source" model. As the name implies, the defining feature of this model is that it allows users to review and in many cases modify and redistribute the human-readable form of software known as source code. Supporters sometimes claim that *the open-source model produces software that is technically equal or even superior to programs developed under the "commercial" model* pursued by most software firms. (Smith, 2002, p. 461)

Section 1. Theoretical Framework

Before approaching the empirical case studies, we introduce our theoretical framework and the methodology of our research. The empirical research presented in this paper comes from two doctoral dissertations based on an emergent and socio-technical approach to the investigation of FLOSS. The focus of the first dissertation was the investigation of the politics and practices of FLOSS licensing (De Paoli, 2008), in the case study of the Geographical Information System known as GRASS (http://grass.osgeo.org/). The focus of the second dissertation was on the relationships between FLOSS socio-technical systems and Freedom as a political concept (Teli, 2008) in the case study of the Operating System Opensolaris (http://hub.opensolaris.org/bin/view/Main/).

Both dissertations were based on a Science and Technolgy Studies (STS) theoretical framework, but in particular on selected aspects of the approach known as Actor-Network Theory, or simply ANT (Latour, 1987 and 2005; Callon, 1986; Law, 1987 and 2004). ANT is an influential STS approach that mixes semiotic, anthropology, constructivism and phenomenology. The limited space of a paper does not allow us to describe all the aspects of ANT. We limit ourselves to a selection of core aspects.

ANT poses emphasis on the G*eneral Symmetry* existing between human and non-human actors. In other words, ANT emphasizes that material reality is composed of hybrid entities (actor-networks or quasi-object) composed of a mixture of humans and non-humans. Agency therefore cannot be attributed to the social actors only or to the material actors only, but rather to hybrids actors. An example (see Latour, 1999) is the *gunman* which is responsible of the action of killing someone: the gunman is an hybrid actor composed of a human (man) and a non-human (gun). The ANT approach to action is opposed to a purely sociological explanation of action – for which only the man will be responsible for the action of killing – or to a purely materialist explanation – for which it is the diffusion of guns in society that lead to people killing people. The General Symmetry of ANT applies well to the investigation of FLOSS development activities that are composed of a mixture of human entities such as users, developers, software companies, public bodies and so on, and non-human entities, for example the source code, the software licenses or the development infrastructure including mailing lists or version control software (see for instance Tuomi, 2001; Lin, 2004; Lanzara and Morner, 2005; De Paoli and D'Andrea, 2008; De Paoli *et al*, 2008; Cornford *et al*, 2010).

A second crucial aspect is the principle that ANT borrows from ethnomethodology whereby the observer-researcher does not decide in advance the social and technical attributes of socio-technological systems (or FLOSS projects in our case). Instead, these attributes can be considered *ethnomethods* (Garfinkel, 1967) that emerge as outcomes of the negotiations among human and non-human actors. Ethnomethods are native conceptions, terminologies, explanations and in general methodologies used by the actors to make sense of the world they inhabit. These native conceptions and methodologies are epistemologically opposed to those of a possible (and fictional) external scientific observer educated in the relevant scientific domain (Lynch, 2007). This implies that the observer/researcher is required to not impose or implement in advance a theory to explain or understand the events under investigation. Michael Callon (1986, pp. 200-201), one of the key authors of ANT, describes this approach as follows: "*the observer must consider that the repertoire of categories which he uses, the entities which are mobilized, and the relationships between these are all topics for actors' discussions. Instead of imposing a pre-established grid of analysis upon these, the observer follows the actors in order to identify the manner in which these define and associate the different elements by which they build and explain their world, whether it be social or natura*l.". Therefore, following Callon, in our investigation we consider the importance of ethnomethods, or how FLOSS stakeholders themselves (including non-human artefacts) account for what they do without imposing our own theory on the events under investigation. In particular it is thanks to this principle that we have been able to observe that the terminology commercial software is a constituent part of FLOSS stakeholders discursive practices.

According to Akrich (1992), one of the key methodological tools for approaching how the actors build and explain their world (ethnomethods) is to focus on the moments of rupture that occur in the "natural flow of things", and in particular on those situations in which devices and technologies go wrong. The author observed that we need to focus on disputes around technological or devices failures as the crucial moments that reveal the actors ethnomethods. Winograd and Flores (1987), in their work on the design of computer artefacts, proposed the specific term *breakdown* in order to capture these moments of rupture. During breakdowns, the objects that populate the world we inhabit and that we take for granted (and that therefore lie unobserved in the background) become present to us as they become the subjects of controversies, negotiations. and adjustments. Indeed, when technological devices breakdown, actors become aware of their presence and, most importantly, actors undertake a series of actions to fix the broken devices. If our laptop stops to work then we will undertake all the necessary action for fixing it. Therefore during breakdowns we, as observers, can be direct witnesses to the actors' efforts to bridge and solve the ruptures and restore order in their social worlds. In other words, the concept of breakdown provides us with a concrete way of approaching the relations between the terminology *commercial software* and FLOSS as ethnomethods.

2. Materials and Methods

In order to investigate the relations and connections between human and non-human entities, and coherently with our theoretical approach, we adopted a methodological perspective compliant with the ANT principle of not imposing a grid of analysis at the beginning of the inquiry (Callon, 1986). Therefore, we conducted our two case studies by the means of qualitative research, following the unexpected (Nardi, 2010) as it was emerging from the empirical field. The majority of the data used in this paper come from the investigation of the GRASS and Opensolaris mailing lists and from an analysis of natural documents, such as software licenses, web pages or technical reports.

In the case study of GRASS most of the events under investigation were located in the past (during the period 1999-2006, whereas the research was conducted between 2005-2008). Therefore, in this case we used an investigation of Mailing Lists archives (the GRASS Users Mailing List Archive[v] – *GUML* hereafter; and the GRASS Developers Mailing List Archive[vi] – GDML hereafter) and other archived documents, in particular archived versions of the GRASS website (retrieved with the web archive http://www.archive.org). The investigation of GRASS lasted for about 24 months and involved the collection and analysis of about 27848 emails organized in 8445 threads for the GUML and 29434 emails organized in 9163 threads for the GDML.

In the case of OpenSolaris the research was conducted as an ethnography exactly during the period in which Sun was undertaking the migration of the proprietary operating system Solaris to FLOSS (also defined as the "opensourcing" of Solaris), between June 2005 and March 2007. The ethnography of OpenSolaris involved the observation of several Mailing Lists, like the general opensolaris-discuss[vii], the governing body cab-discuss[viii] (then renamed ogb-discuss[ix]), the technical opensolaris-code[x] and opengrok-discuss[xi]. Moreover, the English IRC channel of the project, as well as about thirty developers' and managers' blogs were followed.

All the data presented in this paper were analyzed using a Grounded Theory approach (Glaser e Strauss 1967). In Grounded Theory social theory is the outcome of the analysis and the theory in itself is outcome of the recursive relationships between the data (thorough coding) and the concepts composing the theory. Grounded Theory allows the creation of social theory (by an articulation of the concepts) in an inductive way and starting from empirical data. The observations on the use of the terminology commercial software by FLOSS stakeholders hence emerged as an outcome and as a bottom-up theorization of the data analysis.

3. What Does it Mean Commercial Software? The GRASS Case Study

In this paragraph we analyse the relationships between FLOSS and the terminology *commercial* *software* as they emerge directly from software developers practices in the case study of the GRASS project. GRASS born at the beginning of the ‘80s as a small project of the United States Army Corp of Engineering Research Laboratory (USACerl). The system was distributed by the US Army as public domain software and hence as a particular form of FLOSS. The project grew very fast. In 1993, GRASS source code was approximately 300,000 lines, with more than 15 locations developing the system, at a development effort estimated to be the work of five person-years (Westervelt, 2004). In 1996, however, USACerl announced its decision to stop GRASS development (USACerl, 1996). In 1998, a new GRASS Development Team (GDT) was formed with the purpose of furthering GRASS development and creating a new community of users. In particular the new GDT wanted to develop GRASS as a Free and Open Source software. The new GDT included (and still includes) a group of volunteers affiliated to several public and private international organizations. The October 1999 has certainly been one of the milestones of the recent history of GRASS, as the software was released under the terms of the GNU/GPL licence, Version 2 (FSF, 1991)[xii]. The GPL is the main Free and Open Software license used on thousands of software including, among others, the well know operating system GNU/Linux. The GPL is well known for a specific licensing term know as the *Copyleft*. The Copyleft term states that derivative works based on a previous GPL’ed software, must be GPL’ed as well. In other words, the Copyleft clause is hereditary and once the license it is applied to a software it remains on that forever: in this way the license ensures that the source code of software is always available and modifiable – together with the object code – for the public along the chain of distribution of the software (see Weber, 2004).

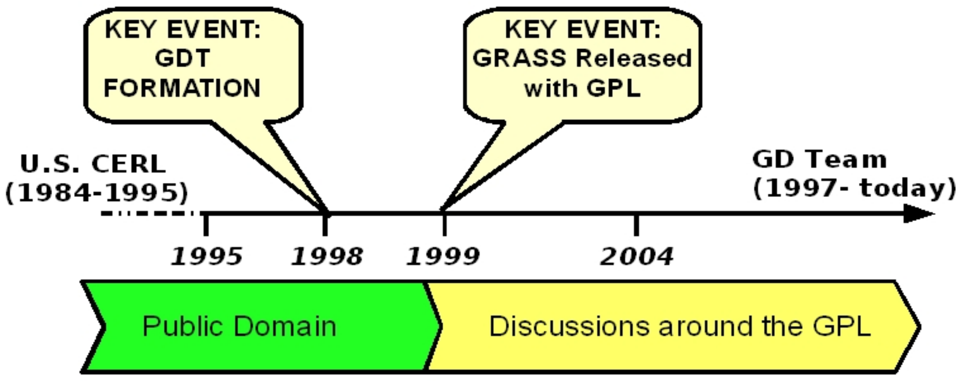

Figure 1 – Key events of the GRASS case study

The license change in the GRASS case study was one of those moments of rupture/breakdown that we identified as crucial for observing ethnomethods that otherwise remain hidden or taken for granted by actors. This is true also for the relationships between FLOSS and the terminology commercial software. Indeed, the fact the GPL was not the original license of GRASS triggered a process of conflicts resolution: the GRASS developers were forced to solve Copyright conflicts between the GPL itself and the licenses of several commands/modules/libraries of GRASS. It is in the resolution of these licenses conflicts that we observed the emergence of the commercial definition of FLOSS.

3.1 GRASS as Commercial Software

File formats are ways of organizing computer data. A common issue with data formats is the existence of both closed and open formats. In the first case the specifications of the organization of data inside the file is kept secret by the producer (common cases is for example the DOC data format) while, in the case of open data format the specifications are fully published[xiii] and open. Common examples of open data formats are the Adobe PDF or the OpenOffice ODT format.

GIS technologies, such as GRASS, use a varieties of Geographical data formats and for this reason import-export functionalities are required to ensure compatibility between different formats.

In this paragraph we describe the conflict between the GPL license and the license of a well known import/export library – the OpenDWG library – used in GRASS for managing the data format known as DWG. DWG is the native, and proprietary, format of several CAD packages including the well known AutoCAD. Therefore, for a GIS managing the DWG format is an important feature as many maps may come in this format.

The library OpenDWG is a software distributed in both binary and source code form. This library was written with the goal of providing a way for manipulating the DWG closed data format by a "*membership-based consortium of software companies, developers and users committed to promoting the open exchange of CAD data now and in the future* " (from http://www.opendwg.org/). This consortium is called Open Design Alliance. This library (OpenDWG) was introduced in GRASS during 2003 as a way to enble GRASS users to use DWG maps, in particular thorough a specific GRASS DWG import command, known as *v.in.dwg*. After the introduction of this library in GRASS, however, the following message was posted on the GRASS Developers Mailing List by a developer:

Noticed that v.in.dwg from GRASS 5.1 [...]

uses the proprietory (*sic*) library opendwg. As I believe that it also needs the GRASS libraries which are under GNU GPL, this means that v.in.dwg has a severe license problem.

[GDML, 13 May 2003, http://www.osgeo.org/pipermail/grass-dev/2003-May/007897.html]

The developer describes the GRASS command *v.in.dwg* as having "*a severe license problem*": this is a moment of breakdown in which the use of *v.in.dwg* which was taken for granted becomes problematic. The problem is that the GRASS command v.in.dwg was using both the OpenDWG library and the GRASS library that at compilation time were becoming a single software and hence a single derivative work of art. The developer pointed out that this was a violation of the GPL at the moment of distribution of GRASS binaries:

We should remove v.in.dwg because nobody can distribute binaries
and if somebody did, this person would violate the license of GRASS
which mean he strictly would loose the right to use GRASS.
[GDML, 13 May 2003, http://www.osgeo.org/pipermail/grass-dev/2003-May/007897.html]

This point of the discussion will become clear below when we will present the controversial aspect of this conflict between licenses. For the moment we should note that the GDT, after this post, took the decision to eliminate the OpenDWG library from GRASS. This decision however lead to a direct result: GRASS would lack any possibility to manipulate maps in the DWG data format and therefore this could introduce a serious limitation on the use of GRASS in comparison with, for example, proprietary GISs.

The debate/discussion around the possible inclusion of the OpenDWG library in GRASS – with hence the possibility to use DWG maps – did not end here. Almost one year after the elimination of OpenDWG library, one of the GRASS developers posted the following message on the Developers Mailing List, describing the terms and conditions of the OpenDWG library:

To my surprise, their web site description of Associate Member terms and conditions permitted the distribution of the libraries in software that is distributed free of charge. This certainly fits GRASS.

[GDML, 26 August 2004, http://grass.itc.it/pipermail/grass5/2004-August/015218.html]

According to this developer, the “status” of Associate Member of the OpenDWG Alliance grants the permission to use and distribute the library in software which are “*free of charge*”. For the developer this is a situation that “*fits GRASS*”. In order to be sure about his claims, the developer called the OpenDWG Alliance Head Quarter, asking for clarifications about the use of the OpenDWG library:

So I called them this morning. I had a good discussion with the membership coordinator with the Open Design Alliance. He assured me that the alliance's intent was only to restrict or control commercial use of their libraries, not use in educational or free software.

[GDML, 26 August 2004, http://grass.itc.it/pipermail/grass5/2004-August/015218.html]

This phone call seemed to clarify any doubt about the possibility to use the OpenDWG with GRASS: the Open Design Alliance license scheme would allow the use of the library in non-commercial software and in Free Software. The following is an excerpt from the OpenDWG *Terms of Use* for the Associate Membership, that was posted on the GRASS Developers Mailing List, and that was followed by a comment from the developer:

> *Dear New Associate Member :*
>
> *I have given you access to the DWG files according to the Associate Member Agreement.*
> *We allow access to our libraries for research purposes and development of free or*
> *internally used software only.*

Unless there is some catch to the GRASS GPL license that I am missing (quite possible, I suppose, given my lack of legal expertise), I think we can distribute openDWG libraries with GRASS as long as we don't sell GRASS commercially--something prohibited by the GPL license.

[GDML, 26 August 2004. http://grass.fbk.eu/pipermail/grass5/2004-August/015218.html]

According to the developer then, it seemed to be possible to distribute GRASS and the OpenDWG library as compiled software as long as GRASS was not sold commercially. This situation, still according to the developer, seemed to be prefigured by the GPL license itself, due its provision of preventing the commercialization of software. Here we witness therefore a possible opposition between a FLOSS software (covered by the GPL) and the terminology *commercial software*. However this interpretation of the GPL license as opposed or antonym to the commercialization of software – which is exactly the one portrayed by the academic literature – was contradicted by another GRASS developer on the Developers mailing list:

The GPL in no way prohibits commercial distribution of software (look at all the GNU/Linux Distributions that sell GPL'd software). Free in the sense of free software (and in the sense of the GPL), does not mean non-commercial, it means the freedom to access, modify and redistribute modified version of the source code. But you have every right to sell GPL'd software, including.

[ML, GDML, 27 August 2004, http://grass.fbk.eu/pipermail/grass5/2004-August/015224.html]

In this message the developer clarifies that the word "Free" as it relates to Free Software does not mean "gratis" and in no way is in opposition to commercial software. As we can see this message clearly pushes a definition of *FLOSS as commercial software* which is opposed to the antonym between FLOSS and commercial software often presented by the academic literature. Indeed, according to this GRASS developer the GPL requires that the software covered by different licenses abide by some restrictions, in order to be compatible with the GPL itself. The Copyleft clause (terms 2b in GPL V2.0) is one of such restrictions (see for a discussion De Paoli *et al*, 2008). Another case is the term number 1 of the GPL which states that it is possible for the users to distribute a software in exchange of a fee/payment[xiv]. In other words, all the software covered by the GPL should be considered as a *specific form of commercial software*, to the extent that developers are entitled by the license to distribute copies of the software asking for a payment. Indeed, this definition (FLOSS as commercial) is also supported by concrete examples that are quoted in the previous message: the GNU/Linux commercial distributions. To make this clear, despite being distributed in both source and object code, the OpenDWG software can be used for commercial purposes only after the payment of a fee, whereas non-commercial use is granted by the license attached to the Open Alliance Membership. Therefore for a user just using the GRASS command *v.in.dwg* on the local computer was not a direct violation of licenses, however for a company that distributes GRASS object code requesting the payment of a fee this is a violation of the GPL and of the OpenDWG license as well.

We can easily understand that what is at stake in this discussion is exactly the definition of what is a commercial software and whether FLOSS can be considered commercial software or not. Commenting the content of the phone call between the first GRASS developer (the one who raised the point that OpenDWG is compatible with the GPL) and the OpenDWG Alliance, another GRASS developer posted the following message:

Michael cited a phone conversation with Aaron Dahlberg of the Open Design Alliance: "He assured me that the alliance's intent was only to restrict or control commercial use of their libraries, not use in educational or free software."

It's debatable, but I would say that Mr. Dahlberg does not know the definition of Free Software and would not have included Free Software in his statement. I am pretty sure he meant proprietary gratis software (aka freeware).

[GDML, 30 August 2004, http://grass.fbk.eu/pipermail/grass5/2004-August/015249.html]

According to this GRASS developer the spokesperson at Open Design Alliance meant to address what is known as freeware rather than Free Software, in other words a type of proprietary software which is distributed free of charge. It is clear therefore that defining Free Software as commercial is a matter of clarifying the terms used to describe things and it is a matter of making order in the world inhabited by developers. In order to clarify the confusion of terms between commercial and proprietary software, the following statement (taken from the OpenDWG website) was posted on the GRASS mailing list:

Open Design Alliance members have created the following free utilities, based on the OpenDWG Libraries, for your unrestricted, non-commercial use. Please note that inclusion of any utility in a commercial product does require commercial licensing[xv]

This post finally clarified to GRASS developers that it was not possible for them to use the OpenDWG Library together with GRASS, due to the commercial nature of GRASS itself granted by the terms of the GPL (for instance by the term 1). Indeed, the OpenDWG source code can be freely used, as it is clearly stated above, but *only for non-commercial purposes*. Therefore we have an opposition not between FLOSS and commercial software, but rather *between FLOSS* and *non-commercial software*.

The idea that FLOSS is not just the opposite of commercial software is part of a clear and well defined strategy of developers (it is indeed an ethnomethod!!) and not just something that is part of academic critique and discussion. Several others examples could be taken from the GRASS case to justify this statement. We propose a further, shorter and revealing example that clarifies that for FLOSS developers the antonym of FLOSS is not commercial software but rather proprietary software. This examples relates with the role of the Open Source GeoSpatial foundation (OSGeo)[xvi] in enhancing and promoting the use of FLOSS GeoSpatial software, including therefore GRASS:

Hello,

as part of our marketing strategies as OSGeo we try to be careful in our

wording. One trap that we try to avoid is opposing Open Source software to "commercial software" as this is not the appropriate antipode to what we are trying to say:
http://en.wikipedia.org/wiki/Commercial_software

The term "commercial" itself can be perfectly applied to Open Source and Free Software:
http://wiki.osgeo.org/index.php/Commercial_Services
http://wiki.osgeo.org/index.php/%22Commercial_Software%22

The opposite to Free Software licensing is proprietary licensing and the opposite to Open Source development methodology is closed source. The distinction here is best formulated as Open Source vs. Closed Source (development wise) and Free Software vs. proprietary (licensing wise).

[7 October 2007, From
http://n2.nabble.com/UN%27s-program..%3A-ESRI-and-cities-mapping-td1879544.html#a1879547]

This message makes clear that for FLOSS developers a clarification on the commercial nature of FLOSS is matter of defining what they themselves are and do. It is a matter of identity for FLOSS communities. Indeed there is a need for clarification about the precise meaning of the term commercial: according to OSGeo spokepersons the commercial nature is an inherent characteristic of FLOSS. OSGeo says that the word commercial can perfectly be applied to FLOSS and that the antonym of FLOSS is what we can define to as proprietary software: this ethnomethod is very much different from what is depicted in the academic literature.

4. Community Building and Commercial Software: The Case of OpenSolaris

The example of GRASS shows that the use of the terminology *commercial software* is part of the everyday development practices in voluntary based FLOSS projects, as a way to establish the *nature* of FLOSS itself or, in other words, to establish what is proper of FLOSS. We can reasonably assume that the involvement of corporations in FLOSS makes it possible to think about the relationships FLOSS and commercial software as part of specific strategic commercial plans. In this paragraph we discuss this point by looking at the case of OpenSolaris.

OpenSolaris is an Operating System born from the release of the proprietary Solaris Operating System with a FLOSS license in 2005 by Sun Microsystems[xvii], at the time one of the major IT players globally. Later in 2010 Sun was acquired by the Oracle Corporation and since then the project OpenSolaris is undergoing a series of deep changes, e.g. a fork[xviii]. Nonetheless, OpenSolaris constituted a major experiment of migration from proprietary software to FLOSS both in technological and organizational terms. Indeed, Sun migration of OpenSolaris involved not only a shift in licensing models but also a shift in software development practices with the need to build a FLOSS community around the system. OpenSolaris constitutes a crucial case study for our understanding of the FLOSS phenomenon and for our goal of linking the terminology *commercial software* to FLOSS.

In the case of OpenSolaris, we can identify two different arenas of action related with the commercial nature of FLOSS: (1) an OpenSolaris-based distribution that Sun provided to its customers; (2) the translation of the pre-existing proprietary and closed software development, into a FLOSS community based software development. In both these arenas, the intersection between the commercial nature of Sun and FLOSS emerged as part of the company strategy. For instance, the definition of OpenSolaris as a commercial project was clearly stated in the first version of the "*OpenSolaris Governance Proposal*"[xix] (later on called "*Constitution*"[xx]) a document that detailed the governance mechanisms of the project. This document, in its first sentence, defined OpenSolaris as "*an organization dedicated to the collaborative production of open source software for a family of commercial-grade operating systems*" (OpenSolaris Governance Proposal, Draft 00, italic emphasis added). In this case, the definition of “commercial-grade operating systems”, later on discharged, was used to identify the quality of Sun engineering technology as consolidated in the commercial product Solaris, as shown in the following part of the “Principles” of OpenSolaris: “*Quality is always a top priority. The OpenSolaris project will continue the long tradition of quality engineering established by the Solaris Operating System (OS)*”.

A key aspect of the OpenSolaris enterprise, was Sun Microsystems' licensing strategy and the licensing rhetoric used by Sun and related to the software industry business. This rhetoric is presented in a book written by two Sun engineers. Goldman and Gabriel (2005, p. 1) stated that: “*business is changing after the expansive thinking of the late 1990s followed by the lessons learned in the early 2000s: It no longer makes sense for every company to make and own every aspects of its business*”. For Sun engineers FLOSS was a fundamental way for conducting business in the contemporary software market in a situation in which *Innovation Happens Elsewhere* according to the title of Goldman and Gabriel book. In other words, business could be seen as a way to harness innovations being developed by others. Hence, the authors articulated their vision of “making FLOSS a business practice” as a “vision of community building” focused on enrolling innovators located outside the boundaries of the company in itself: this created fundamental connections between a specific aim (the business) and a set of non-human artefacts designed to increase the participation of a range of entities (the community building: including both stakeholders and other software), different from Sun Microsystems, its employees, and its technology. Specifically, the artefacts meant to increase the participation in OpenSolaris community were: (1) the software licenses used by Sun and (2) the infrastructure supporting the development. Here we will focus on OpenSolaris license.

The license, as in the case of GRASS, is a key non-human artefact for understanding the commercial dimension of OpenSolaris. The license chosen by Sun Microsystems for the release of OpenSolaris codebase was a brand new copyleft license, the *Common Development and Distribution License* (CDDL). This license was prepared by Sun taking inspiration from another

Open Source License the Mozilla Public License (MPL). The CDDL was written with the aim of building a network of entities around OpenSolaris different from that enacted by other licenses such as for example the GPL (see De Paoli *et al*, 2008 for a comparison between the two licenses). Sun choose a file-based license (on the model of the MPL) covering each single file of the system rather than the whole system at once, therefore a license different from a program-based licenses such as the GPL. The specific goal of this choice was stated in the first CDDL Frequently Asked Questions [xxi] (FAQ) (italic emphasis added):

“We wanted a copyleft license that provided open source protections and freedom and also enabled creation of larger works for commercial purposes. “

We clearly see that both open source protections and commercial purposes are here presented in the same sentence as two complementary aspects of Sun licensing and community building strategy, and clearly not as two terms in opposition. Again this use is very different from that we often read in literature. One of the aim of Sun was to allow the distribution of OpenSolaris code files, still protected by the CDDL, together with proprietary code in a software distribution sold for a fee. The most notable example was the same Solaris proprietary version, distributed together with Sun's hardware or via a website (without support), and regulated by a traditional proprietary software Software License Agreement[xxii].

The modifications that Sun did for adapting the MPL license in the creation of the CDDL are an important aspect for understanding the commercial nature of the project. One of the terms of the MPL which was removed by Sun was the 8.2(b) and the reason why Sun decided to remove this term was as follows (italic emphasis added):

> We removed MPL's 8.2(b), which revoked license rights if patent claims are made against any product of a Participant, not just code released under this license. We're trying to build a community of diverse

> contributors, large and small, including *commercial contributors*, and felt that this section would be a hindrance to *commercial adoption*.

“Any Product” in the above statement, and according to the original MPL term, meant not only software code but also and in particular any “hardware, or device” covered by a patent[xxiii]. This term was considered an obstacle in enrolling commercial entities especially due to the limitations imposed on hardware. The crucial point of the previous Sun comment to the MPL 8.2(b) term relates again with the community building effort and the focus on enrolling also commercial contributors: actors interested to use OpenSolaris in their business activities or, in other words, what in the FAQ is defined to as the commercial adoption of OpenSolaris.

A further CDDL FAQ clarifies another important aspect of this link between the FLOSS and the commercial dimensions of OpenSolaris:

> May I use the OpenSolaris source code or binaries commercially?
>
> Yes, you may use the OpenSolaris source code in commercial products. Note that if you distribute binaries built from code released under the CDDL, you will need to meet the terms of the CDDL and distribute the corresponding source code under the CDDL. See the license for details.

As explained in this FAQ the CDDL license allows to use both the source and the binary code of OpenSolaris commercially, and not just the binaries as in proprietary distributions. The commercial dimension of OpenSolaris here is clearly not referred to proprietary software as the answer to the FAQ makes clear. Any commercial distribution of binary code derived from CDDL code must include always the distribution of the source code of the released binaries to comply with the copyleft provisions of the CDDL. This makes extremely clear that the terminology commercial products here does not mean necessarily proprietary software distributed only in binary form. However, in cases in which CDDL code is used together with proprietary software, the owner is not required to release the code of proprietary software. This point is again made clear in one of the

CDDL FAQ, which implicitly states that it is the terminology proprietary software which stand in opposition to the commercial distribution of source code together with binaries under the CDDL :

> If I use code licensed under the CDDL in my proprietary product, will I have to share my source code?
>
> Yes, for any source files that are licensed under the CDDL and any modifications you make. However, you don't need to share the source for your proprietary source files.

An interesting example that illustrates the content of this FAQ is the case of Nexenta[xxiv], a company who develops an operating system that mixes elements of both GNU/Debian and OpenSolaris. Nexenta has a business model defined “Commercial Open Source” or OpenCore Model (Gulecha, 2009). This model is based on the open core OpenSolaris plus GNU/Debian that is extended by proprietary add-ons.

From the documents we have analysed and presented so far it is clear that OpenSolaris is a commercial initiative which clearly does not stand in opposition to FLOSS. The OpenSolaris business, focused on community building, was heavily based on a commercial use of FLOSS software and on the enrolment of commercial entities in the community. What is more important is that this commercial aspect was discursively constructed in Sun documents, which were trying to delimit the boundary of the community building by enhancing the commercial nature of the project and the commercial interests of those enrolled in the project.

However, it is important to disclose that the constituent commercial nature of OpenSolaris is less clear as in the case of GRASS. This is quite surprising and it seems that in volunteers FLOSS project the commercial nature of FLOSS needs to be defended more heavily. Indeed, in OpenSolaris Mailing list discussions the terminology commercial software is often portrayed by participants as an antonym to FLOSS. This is something that rarely happened in the case of

GRASS. For instance, Goldman and Gabriel (2005) the Sun engineers mentioned at the beginning of this paragraph, used the word commercial as an antonym to FLOSS in their books:

> If you still don't believe that open-source software is of similar quality to most commercial software, just take a look at some open-source software you use every day. (p. 47)

These considerations do not undermine the original thesis of this paper, but show however that reducing the boundary between the terminology commercial software and FLOSS is more crucial for developers in volunteer based FLOSS projects rather than in corporate FLOSS projects for which the commercial aspect is taken-for-granted as a starting point for subsequent practices of detailed definition and legitimization.

5. Discussion and Conclusion

We began this paper by stating how in mainstream FLOSS literature there is often an opposition between FLOSS and the terminology commercial software. The problem of this paper was not to state that FLOSS is also a commercial initiative or a business activity (Perens, 2005), but rather to show that using the terminology commercial software as an antonym to FLOSS seems not be what is happening in the field. In fact, with our empirical, qualitative and bottom-up analysis of the GRASS and OpenSolaris cases we showed that this opposition between FLOSS and commercial software is not grounded in current development practices and discourses.

According to the Merriam Webster Online Dictionary “commerce” can mean at least two different things: first, it is a “*social intercourse*” involving the “*interchange of ideas, opinions, or sentiments*”; second, it is “*the exchange or buying and selling of commodities on a large scale involving transportation from place to place*”. Therefore commercial practices are double – sided: on one side, they are the interchange of ideas and opinions, on the other side they involve the commodification of products and their trade. It is interesting to see that especially the first definition of “commerce” seems to adapt well to the practices of FLOSS, probably even more than to proprietary and closed source software.

The crucial aspect of our framework are FLOSS developers ethnomethods (and FLOSS stakeholders in general) and the way stakeholders make order in the socio-technical worlds they inhabit. For instance our analysis showed that in the case of GRASS, the commercial character of FLOSS is clearly linked with the provisions of the GNU GPL (version 2.0 and in particular with the term 1) that allow the distribution of copies of the software for a fee. According to the developers, therefore, GRASS is a commercial software exactly because it is covered by the GPL. At the same time the commercial nature of GRASS is used by the developers as a strategic element that can help to enrol more developers in the community. In addition, GRASS developers strongly oppose themselves to the use of the terminology commercial software to characterize only proprietary software. In the case of OpenSolaris instead the links between commercial software and FLOSS are part of a community building process whose aim is to enable the commercial use of the OpenSolaris code as well as facilitating the enrolment of commercial innovators in the community. In both cases the commercial side of FLOSS seems to be constructed by Sun via the licensing strategy of the CDDL.

A question at this point arises: why academic literature opposes FLOSS to the terminology commercial software, despite being clear that for developers they are not always in opposition? We try, in the remaining of the conclusion, to answer this question with a reflection prompted by the work of the master of Science-Fiction Philip K. Dick. In his introductory essay to the collection of short stories *I Hope I Shall Arrive Soon & Other Stories*, Dick (1987) discussed two recurring themes in his work: “what is the real man?” and “what is reality?”. It is the second of these themes that is of particular interest to us. The phrase we would like to quote is the following: “*The basic tool for the manipulation of reality is the manipulation of words. If you can control the meaning of words,* *you can control the people who must use the words.* " (Dick, 1987). In the essay Dick argued that reality is not something that is out-there, ready to be discovered or used. Rather, for Dick reality appears to be something that is "manufactured" along with the way we act in it. This vision of reality as something constructed by our actions, rather than something that is simply given, even lead him to consider that we should speak not just about reality, in singular terms, but about realities in plural terms.

This view on reality is very close to that of Actor-Network Theory (ANT) and Social Constructivist approaches in which realiti(es) are seen as the result of construction processes that involves an array of human and non-human elements (Latour, 1999), deeply interlinked in heterogeneous networks of power (Latour, 1987; Law, 1987). Dick's argument on realities, often referred to as *ontological politics* in ANT (Mol, 2002), is more suitable for the conclusion of this paper because it helps us to emphasize the role of words in this realities construction process. Indeed, what Dick seems to argue is that the manipulation and control of words is not just a neutral process that does not influence reality. Words are not just neutral elements that simply represent the things that compose reality out-there. Rather, words are the bricks we use to "manufacture" and sustain a specific definition of reality. Therefore, the manipulation and the control of a certain set of discursive practices is also related to the ability to manufacture a certain reality: discourse and power are deeply interleaved (Foucault, 1970).

This brief digression in Dick's view about reality helps us in better framing the problem of this paper: the relationship between the terminology commercial software and the phenomenon of FLOSS, given that the academic literature often assumed them as an antonym. What we described in this paper is the process through which the use of the terminology *commercial software* in relation with FLOSS and proprietary software is subject to control dynamics. Often (this is very clear in the case of GRASS) FLOSS developers want to free the terminology commercial software from the control of specific discursive practices and discursive uses. This because the terminology commercial software is meant, by FLOSS developers, to build a specific definition of FLOSS in which the antonym of FLOSS is not commercial but proprietary software. By contrast, in the GRASS case it appears that it is the proprietary world that somehow seeks to control the use of the world commercial either in the text of software licenses or in generic discussions. In other terms, from FLOSS developers point of view, it seems that the proprietary world wants to manufacture a reality in which FLOSS is portrayed as the antonym of commercial software. In the case of OpenSolaris instead, Sun was actively trying to take control of the terminology commercial software in order to pursue its community building strategy. By pushing a specific definition of community building as a commercial enterprise, Sun goals was that of enrolling commercial innovators into the OpenSolaris community. A clear outcome of both case studies therefore is that for developers controlling the meaning of the terminology commercial software is a process for manufacturing FLOSS reality.

At this point there is an even more important issue that we need to emphasise: the role of academic discourse. It seems that scientific publications often uncritically assume FLOSS as an antonym of commercial software. Given the above considerations about the role of words in building FLOSS reality, we should reflect on whether the academic discourse participates to a specific construction of reality, rather than being the manifestation of a supposedly *neutral* point of view. We should ask ourselves which reality we contribute to manufacture by being academic and researchers of FLOSS. As we showed in our literature review, several important academic contributions fully contribute to a definition of reality in which FLOSS is sharply opposed to commercial software and in which proprietary software is often portrayed as synonym of commercial software. In his essay Wheeler (2006) argued that this is just a confusion and a mistake, because writers are "s*imply unable to understand what is happening in the software industry*". Again, Philip Dick vision on realities and the use of words can help us here. We can argue that the way the word commercial is used by the academic discourse around FLOSS might not just be a simple misunderstanding, but rather it reflects the control over the use of words itself. By describing FLOSS as the opposite the commercial software and by the describing proprietary software as synonym of commercial software, the academic debate helps in manufacture a reality which stand in opposition to that manufactured by FLOSS stakeholders. In order to avoid this problem we think that the direction to take is conducting empirical research that is informed by an ethnomethodological view of FLOSS stakeholders. Above all ethnomethods are native conceptions that are epistemologically opposed to those of a possible (and fictional) external scientific observer educated in the relevant scientific domain (Lynch, 2007). Our investigation showed that FLOSS developers ethnomethods (namely that FLOSS is commercial in both its nature and processes) depicts a situation radically different to that described in a large body of academic literature.

About the Authors

Stefano De Paoli[xxv] ***-*** Is Research Fellow at the Foundation <AHREF in Trento (Italy). Stefano holds a PhD in Sociology and Social Research, with specialization in Information Systems. He has conducted research on Free Software Licensing, Massively Multiplayer Online Games and the Future of the Internet. Stefano.depaoli@gmail.com

Maurizio Teli - PhD in Sociology and Social Research, University of Trento (Italy), has a background in Political Science. He is involved in and researches about the importance of FLOSS “practices of freedom” in the processes of organizing a community and producing technology. maurizio@maurizioteli.eu

Vincenzo D'Andrea - is an associate professor at the University of Trento, where he teaches Information Systems. His research interests includes service–oriented computing, free and open source licensing, virtual communities. Vincenzo.dandrea@unitn.it

References

M. Akrich, 1992. ”The De-Scription of Technical Objects,” In: W. Bijker and J. Law (editors). *Shaping Technology, Building Society: Studies in Sociotechnical Change.* Cambridge, Mass: MIT Press, pp. 205-224.

Bessen, James, 2005. “Open Source Software: Free Provision of Complex Public Goods. Research on Innovation paper” http://researchoninnovation.org/opensrc.pdf. Accessed 10 June 2009.

Bird Christian, Nagappan Nachiappan, Devanbu Premkumar, Gall Harald and Murphy Brendan, 2009. “Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista,” *Communications of the ACM*, volume 52, number 8, pp. 85-93.

Bitzer Jürgen, 2004. “Commercial versus open source software: the role of product heterogeneity in competition,” *Economic Systems*, volume 28, number 4, pp. 369-381.

Bonaccorsi Andrea and Rossi Cristina, 2002. “Why Open Source software can succeed,” *Research Policy*, volume 32, number 7, pp. 1243-1258 and at. http://linkinghub.elsevier.com/retrieve/pii/S0048733303000519, accessed 10June 2009.

M. Callon, 1986. “Some elements of a sociology of translation: domestication of the scallops and the fishermen of Saint Brieuc Bay," In: J. Law (editor). *Power, Action and Belief: A New Sociology of Knowledge?*. London: Routledge, pp. 196-22

Chengalur-Smith Indushobha, Sidorova Anna, and Daniel Sherae L. 2010. "Sustainability of Free/Libre Open Source Projects: A Longitudinal Study," *Journal of the Association for Information Systems*: volume 11, number 11, Article 5. http://aisel.aisnet.org/jais/vol11/iss11/5 Accessed 10 January 2010.

Cornford Tony, Shaikh Maha and Ciborra Claudio, 2010. "Hierarchy, Laboratory and Collective: Unveiling Linux as Innovation, Machination and Constitution," *Journal of the Association for Information Systems,* volume 11, number 12, Article 4, at http://aisel.aisnet.org/jais/vol11/iss12/4, accessed 27 December 2010.

S. De Paoli, 2008. *Software, Copyright e Pratiche di Licensing.* Unpublished Doctoral Dissertation, Licensed by the University of Trento (Italy), 04 March 2008.

De Paoli Stefano and D'Andrea Vincenzo, 2008. "How artefacts rule web based communities: practices of Free Software Development," *Int. J. Web Based Communities*, volume 4, number 2, pp 199–219.

De Paoli Stefano, Teli Maurizio and D'Andrea Vincenzo, 2008. "Free and open source licenses in community life: Two empirical cases," *FirstMonday*, volume 13, number 10 (october).

http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/2064, accessed 10 June 2009.

P. Dick, 1987. *I Hope I Shall Arrive Soon*. New York: Saint Martin’s Press.

Free Software Foundation, 2004. “The Free Software Definition,” at http://www.gnu.org/philosophy/free-sw.html, accessed 10 June 2009.

Free Software Foundation. 1991. “GNU/ General Public License License 2.0,” at http://www.gnu.org/copyleft/gpl.html, accessed 10 June 2009.

M. Foucault, 1970. *The Order of Things.* London: Tavistock.

H. Garfinkel, 1967. *Studies in ethnomethodology* . Englewood Cliffs, NJ: Prentice-Hall.

Glaser Bernard G. and Strauss Anselm L., 1967. *The discovery of grounded theory.* Chicago: Aldine.

Goldman Ron and Gabriel Richard P., 2005. *Innovations Happens Elsewhere. Open Source as Business Strategy*. San Francisco: Elsevier.

GRASS Development Team, 1999-2006. “*GRASS History,”* at http://grass.itc.it/devel/grasshist.html, accessed 19 June 2009.

Gulecha Anil, 2009. “Nexenta, Open Storage and Commercial Open Source,” Presentation at *OpenSolaris Developer Conference* 28-30 October, 2009, Dreseden, Germany, http://www.osdevcon.org/2009/slides/nexenta_openstorage_anil_gulecha.pdf, accessed 10 January 2011.

Koch Stefan and Schneider Georg 2002. ”Effort, co-operation and co-ordination in an open source software project: GNOME,” *Information Systems Journal*, volume 12, number 1, pp. 27-42, and at http://www3.interscience.wiley.com/cgi-bin/fulltext/118925737/HTMLSTART, accessed 10 June 10 2009.

Ke, Weiling and Zhang, Ping 2010. “The Effects of Extrinsic Motivations and Satisfaction in Open Source Software Development,” *Journal of the Association for Information Systems,* volume 11, number 12, Article 5. at http://aisel.aisnet.org/jais/vol11/iss12/5, accessed 27 December 2010.

Kogut Bruce M. and Metiu Anca, 2001. “Open Source Software Development and Distributed Innovation,” *Oxford Review of Economic Policy,* volume 17, number 2, pp. 248-264.

Healy Kieran and Schussman Alan 2003. “The Ecology of Open-Source Software Development,” at http://opensource.mit.edu/papers/healyschussman.pdf, accessed 10 June 2009.

Hertel Guido, Niedner Sven and Herrmann, Stefanie 2003. “Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel,” *Research Policy,* volume 32, number 7, pp. 1159-1177,

Lanzara, Giovan Francesco and Morner Michele 2005. “Artifacts rule! How organizing happens in open source software projects,” In: B. Czarniawska and T. Hernes (editors). *Actor-Network Theory and Organizing.* Liber e Copenhagen Business School Press. pp. 67-90.

B. Latour 1987. *Science in Action*. Cambridge Mass: Harvard University Press.

B. Latour 1999. *Pandora’s Hope.* London: Harvard University Press.

B. Latour 2005. *Reassembling the Social. An Introduction to Actor-Network-Theory*. New York: Oxford University Press.

J. Law, 1987. “Technology and heterogeneous engineering: The case of Portuguese expansion,” In: W. E. Bijker, T.P. Hughes, and T.J. Pinch (editors). *The social construction of technological systems: New directions in the sociology and history of technology*, Cambridge Mass: MIT Press, pp. 111-134.

Law 2004. *After Method: Mess in Social Science Research.* London: Routledge.

Lin Yuwei, 2004. “Epistemologically multiple actor-centered systems: or, EMACS at work!,” *Ubiquity,* volume 5, number 1, at: http://www.acm.org/ubiquity/views/v5i1_lin.htm, accessed 27 December 2010.

Lin Yuwei. 2005. The future of sociology of FLOSS. FirstMonday, Special Issue #2: Open Source, October 2005 and at http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1467/1382, accessed 10 June 2009.

Lynch, M. 2007. “The origins of ethnomethodology”. Pp. 485- 516, in *Philosophy of anthropology and sociology* , edited by S.P. Turner & M.W. Risjord. Amsterdam: Elsevier.

Mockus, A., Fielding, R. T., and Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Trans. Softw. Eng. Methodol.* volume 11, number 3 (Jul. 2002), p. 309-346.

A. Mol 2002. *The Body Multiple, Ontology in Medical Practice*, Duke University Press, Durham NC.

B. Nardi, 2010. *My Life as a Night Elf Priest: An Anthropological Account of World of Warcraft*. Ann

Arbor: University of Michigan Press.

Perens, Bruce 2005. “The Emerging Economic Paradigm of Open Source,” *First Monday* Special Issue 2.URL: <http://www.firstmonday.org/issues/special10_10/perens/index.html>. [retrieved June 10th, 2009]

Raymond Eric S. 1999b. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol, California: O'Reilly and Associates.

Smith, B. L. (2002). The Future of Software: Enabling the Marketplace to Decide. In R. W. Hahn (Ed.), *Government Policy toward Open Source Software* (pp. 69-85). Washington, DC: Brookings Institution Press

Sojer, Manuel and Henkel, Joachim (2010) "Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments," *Journal of the Association for Information Systems,* volume 11, number. 12, Article 2. http://aisel.aisnet.org/jais/vol11/iss12/2

Teli, Maurizio (2008). *Libertà e Pratiche. Il software libero e open source nel caso OpenSolaris*. Unpublished Doctoral Dissertation, University of Trento (Italy)

Teli, Maurizio, Pisanu Francesco and Hakken David 2007. “The Internet as a Library-of-People: For a Cyberethnography of Online Groups,”*Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, volume 8, number 3, Art. 33, http://nbn-resolving.de/urn:nbn:de:0114-fqs0703338

Tuomi, Ilkka. 2000, “Internet, innovation, and open source: Actors in the network,” *Firstmonday* volume 6, number 1, http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/824/733 accessed 10 June 2009.

US Army CERL 1996. “Announcements,” http://web.archive.org/web/19970619195255/www.cecer.army.mil/announcements/grass.html, accessed 28 December 2010.

Von Hippel, Eric, and Von Krogh Georg, 2003a. “Special Issue on Open Source Software Development,” *Research Policy,* volume 32, number 7, pp. 1149-1157, http://linkinghub.elsevier.com/retrieve/pii/S0048733303000544, accessed 10 June 2009.

Von Hippel, Eric and Von Krogh Georg 2003b. “Open source software and the" private-collective" innovation model: Issues for organization science,” *Organization Science,* volume 14, number 2, pp. 209-223.

S. Weber, 2004. *The Success of Open Source*. Cambridge Mass: Harvard University Press.

Westervelt, Jim 2004. “GRASS roots. In Proceedings of the FOSS/GRASS users conference,” Bangkok, Thailand, 12–14 September, 2004.

Wheeler, David A. 2006. “ “Commercial” is not the opposite of Free-Libre/Open Source Software (FLOSS),” http://www.dwheeler.com/essays/commercial-floss.html, accessed 23 March 2009.

Winograd Terry. and Flores Fernando 1986. Understanding Computer and Cognition. Norwood, NJ: Ablex.

[i] What we are discussing here in not the FLOSS economic paradigms and FLOSS business models that have been quite widely investigates (see Perens, 2005 and Weber, 2004 for a discussion), but rather the meaning of the word commercial as opposed to FLOSS.

[ii] The software known as Freeware is proprietary software distributed gratis, usually for a limited period of time (trial period).

[iii] Microsoft Windows 7, Adobe Acrobat Reader, Solaris, and OpenSolaris are trademarks or registered trademarks of their respective owners.

[iv] http://aisel.aisnet.org/jais/vol11/iss12/

[v] http://www.osgeo.org/mailman/listinfo/grass-user

[vi] http://www.osgeo.org/mailman/listinfo/grass-dev

[vii] http://mail.opensolaris.org/mailman/listinfo/opensolaris-discuss

[viii] http://mail.opensolaris.org/pipermail/cab-discuss

[ix] http://mail.opensolaris.org/mailman/listinfo/ogb-discuss

[x] http://mail.opensolaris.org/mailman/listinfo/opensolaris-code

[xi] http://mail.opensolaris.org/mailman/listinfo/opengrok-discuss

[xii] A version 3 of the GPL has been released in 2008 by the Free Software Foundation. The event described in this paper are prior the release of this new version of the license. Therefore when we mention the license GPL this refers to the Version 2.

[xiii] The division between open and closed format does not mirror the division between Free and proprietary software. In fact many open data format are realised by proprietary software companies, such as for example the well known Adobe PDF.

[xiv] In this case the GRASS developers are discussing abour the GPL V.20. However, the same terms is present in the GPL v.3.0: term is the number 4. and says "You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee. "

[xv] See http://www.opendesign.com/downloads/guest.htm

[xvi] In particular OSGeo is the recently founded Open Source Geospatial foundation (www.osego.org), an umbrella foundation that gathers several FLOSS project that have in common their geospatial nature. GRASS is one of the founding members of the foundation.

[xvii] The story of OpenSolaris is quite well known and won't be retold here. An interesting and quite descriptive story of OpenSolaris can be read here: http://linux-kertosono.blogspot.com/2010/10/history-of-opensolaris.html

[xviii] Project Illumos (http://www.illumos.org/), a fork of the OpenSolaris project was launched on August, 3rd 2010.

[xix] Retrieved from http://mail.opensolaris.org/pipermail/cab-discuss/2005-July/000763.html

[xx] Retrieved from http://www.opensolaris.org/os/community/ogb/governance/

[xxi] CDDL FAQ, Retrieved at http://openmediacommons.org/CDDL_FAQs.html Accessed 10 January 2010

[xxii] http://www.sun.com/software/solaris/licensing/sla.xml

[xxiii] See the MPL text: http://www.mozilla.org/MPL/MPL-1.1.html

[xxiv] http://www.nexenta.org/

[xxv] Dr. De Paoli is first author of this manuscript.