Why free software is not the antonym of commercial software: two case studies from corporate to volunteer based projects

Stefano De Paoli
Vincenzo D'Andrea
Maurizio Teli

WHY FREE SOFTWARE IS NOT THE ANTONYM OF COMMERCIAL SOFTWARE: TWO CASE STUDIES FROM CORPORATE AND VOLUNTEER BASED PROJECTS

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. Using an approach inspired by ethnomethodology, the paper illustrates how FLOSS developers, both volunteers and corporate employees, use the term "commercial software" as a constituent part of their discourses. We look closely at FLOSS discursive practices uncovering the interactional function of the term "free software" with the aim of provoking a reflection on how such terminology is used in FLOSS academic literature. In particular, we propose examples taken from two case studies: the Geographical Information System known as GRASS and the Operating System known as OpenSolaris.

Keywords:
free and open source software, commercial software, proprietary software, ethnomethodology, actor-network theory

Stefano De Paoli, Vincenzo D'Andrea and Maurizio Teli

INTRODUCTION: COMMERCIAL OR PROPRIETARY SOFTWARE?

In defining what Free/Libre Open Source Software (FLOSS) is, the academic literature has often opposed the term FLOSS to *"commercial software"*. This opposition implies to a certain extent that the term *"commercial software"* is a synonym of *"proprietary software"*. Contrary to the academic literature, we believe that the terms FLOSS and "commercial software" are closely linked. In other words, we show how the antonymous use of the terms "commercial software" and "FLOSS" in academic literature differs from the ways in which FLOSS stakeholders, and especially developers, use the word "commercial" to explicitly refer to what they do. As such, the problem of this paper is not to state that FLOSS is also a commercial initiative or a business activity, something on which today there is quite a large agreement (Perens, 2005) but to highlight the discrepancy that exist between the way the term is used in academic and FLOSS circles. 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 FLOSS stakeholders and, in particular, developers. This paper contributes to the field by offering a greater awareness of how FLOSS actors define what they do, and invites researchers to pay closer attention to the point of view of such actors.

We are aware that there might be different contextual uses of the word "commercial" as it is employed in literature as well as different shades of meanings in the opposition between commercial software and FLOSS. We believe, however, that this possible difference of meaning does not undermine the problematic opposition between FLOSS and "commercial software" presented as self-evident in academic literature.

FLOSS has an approach to software licensing that provides the users with the ability to change and share 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 binary code of software. This, according to the Open Source Definition (2006), does not exclude the possibility of mixing

open source with closed source code in the creation of derivative works. There is therefore a variability in licensing schemes that falls under the FLOSS label and different licensing schemes that FLOSS communities use to protect their work (O'Mahony, 2003): from pure Free Software, usually released under the well know GNU General Public License (GPL), to Open Source software that could also be mixed for the creation of derivative works with closed code, as in the case of software released under the Berkeley Software Distribution (BSD) License.
The GPL is the main FLOSS license used in thousands of software including, among others, the well know operating system GNU/Linux. The GPL has became popular for a specific licensing term known as the *Copyleft*, which ensures that the source code will remain open along the software distribution chain. The Copyleft clause is hereditary and once the license is applied to a software it remains on that software forever: in this way the license ensures that the source code of the software – together with the object code – is always modifiable and available to the public along the software distribution chain (see Weber, 2004). More permissive licenses (Lerner and Tirole, 2005) such as the BSD are non-copyleft licenses and do not require necessarily the distribution of derivative works in both binary and source code with the same BSD license. Further, derivative works originally based on BSD licensed code can also be released in object code only.

FLOSS could – in an ideal typical fashion – also be seen as 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.

The definition of FLOSS should be considered in opposition to the so-called *proprietary* or *closed source software model*. The latter is a software development and business model in which the producer sells the copies of the software binary code in exchange of money [1] (in other words he sells a license that allows the user the execution of the object code). Further, proprietary software is developed exclusively and in full secret by the producer's employees, there is an exclusive control of the software source code which is not accessible to the users. Notable examples of proprietary software are the operating system Microsoft Windows® or the Adobe Acrobat Reader® [2].

In Table 1 below we report a number of literature cases in which such opposition between FLOSS and commercial software is presented. This also includes most of the papers published in the quite recent Special Issue of the Journal for the Association of Information Systems [3] on FLOSS (December 2010), a clear sign of how topical is the issue. Moreover Table 1 shows that the opposition between FLOSS and *"commercial software"* is grounded in different aspects of the FLOSS debate, showing therefore slightly different uses and shades of meanings of the word "commercial", including:

1. The problem of different licensing schemes, with therefore an opposition between commercial software and the openness of the code granted by FLOSS licenses;
2. The different organizations of software development, with an opposition between commercial software development organization and FLOSS development organization;
3. The differences in source code access: open for FLOSS users, closed for proprietary software users.
4. The (broader) efficiency of FLOSS software compared to that of commercial software; and
5. Finally, from an historical viewpoint, opposing commercial software and FLOSS as historical entities.

Our critique of the use of the term “commercial software” – in its various degrees of meaning – as an antonym to FLOSS is based on a simple consideration: the literature does not take into account the meaning and the use of the term commercial software for FLOSS stakeholders themselves, and in particular for developers. On this specific aspect we challenge the mainstream literature by arguing that FLOSS stakeholders use the term “commercial software” to define what they do, as part of well defined and concrete strategies of software development and community building. In this sense, we agree with Wheleer (2006) for whom the opposition between FLOSS and commercial software is not only imprecise, but also mistaken for several reasons:

(a) the rise in commercial development and support for FLOSS, (b) most FLOSS projects' goal to incorporate improvements (which are actually a form of financial gain), (c) official definitions of “commercial item” that include FLOSS, and (d) 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 an investigation that highlights how the term “commercial software” is used by FLOSS stakeholders to construct the 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); and we conclude with a final discussion of findings.

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). Over the past decade Free/Libre Open Source Software (FLOSS) has become a viable alternative to proprietary commercial computer programs. Fuelled 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) 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) 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)

1. THEORETICAL FRAMEWORK

Before approaching the empirical case studies, we introduce the methodology of our investigation. The empirical research presented in this paper comes from two in-depth qualitative case studies. The firts one is the case study of the Geographical Information System known as GRASS (http://grass.osgeo.org/), and the second is the case study of the Operating System Opensolaris (http://hub.opensolaris.org/bin/view/Main/).
Both case studies have been investigated using an ethnomethodological approach borrowed from the research perspective known as Actor-Network Theory (Latour, 1987 and 2005; Callon, 1986; Law, 1987 and 2004). In this paper we emphasize the ethnomethodological perspective of the Actor-Network Theory, rather than drawing on specific concepts such as the general symmetry or the concept of actant. In particular, we base our research on the ethnomethodological principle whereby the observer-researcher should follow and account for actors’ ethnomethods (Garfinkel, 1967). 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 not to impose in advance a theory to explain or understand the events under investigation (Callon, 1986). We begin to see here how ethnomethodology can help shed new light on the problem of this paper: the use of the word “commercial” done by FLOSS stakeholders is an ethnomethod that is epistemologically different from that of academic authors. Therefore, in our investigation we consider the importance of FLOSS stakeholders’ ethnomethods themselves – including how they define FLOSS and “commercial software” – without imposing our theoretical perspective. In particular, through ethnomethodology we have been able to observe that FLOSS stakeholders often consider the term “commercial software” as a constituent part of FLOSS.

According to Akrich (1992), one of the key methodological tools for approaching actors’ ethnomethods is to focus on the breakdowns that occur in the “natural flow of things”, and in particular on those situations in which devices and technologies go wrong. The author observes that we need to focus on disputes around technological failures as the crucial moments that reveal the actors’ ethnomethods. Winograd and Flores (1986), in their work on the design of computer artefacts, propose 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 often 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. When technological devices breakdown, actors become aware of their presence and, most importantly, they undertake a series of actions to fix them. Therefore during breakdowns we, as observers, can be direct witnesses of the actors' efforts to restore order in their social worlds. In FLOSS research, Coleman (2005) and Kelty (2011) have pointed out how crisis – what we refer to as breakdowns – are a privileged point of view for understanding the cultural construction of FLOSS.

Coherently with our theoretical approach, we adopted a methodological perspective that is compliant with the principle of not imposing a grid of analysis at the beginning of the inquiry (Callon, 1986). As such we conducted two case studies guided by the methods and approaches illustrated so far. The majority of the data used in this paper comes from the investigation of the GRASS and Opensolaris mailing lists and from an analysis of natural documents, such as software licenses, web pages and technical reports.

In the GRASS case study most of the events under investigation are located in the past (during the period 1999-2006, whereas the research was conducted between 2005-2008). In this case we researched Mailing Lists archives (the GRASS Users Mailing List Archive [4] – *GUML* hereafter; and the GRASS Developers Mailing List Archive [5] – 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.

The case of OpenSolaris, is an ethnographic research that was conducted between June 2005 and March 2007, during this same period Sun was undertaking the migration of the proprietary operating system from Solaris to FLOSS (also defined as the "opensourcing" of Solaris). The ethnography of OpenSolaris involved the observation of several Mailing Lists, like the general opensolaris-discuss [6], the governing body cab-discuss [7] (then renamed ogb-discuss [8]), the technical opensolaris-code [9] and opengrok-discuss [10]. We also looked at the English IRC channel of the project, and we followed about thirty developers' and managers' blogs.

All the data presented in this paper are analysed 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 the 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 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.

2. WHAT DOES COMMERCIAL SOFTWARE MEAN? THE GRASS CASE STUDY

GRASS was 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 a public domain software. The project grew very fast, and in 1993, the 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 its development. The new GDT included (and still includes) a group of volunteers affiliated to several public and private international organizations. In October 1999, GRASS was released under the terms of the GNU/GPL licence, Version 2 (FSF, 1991) [11].

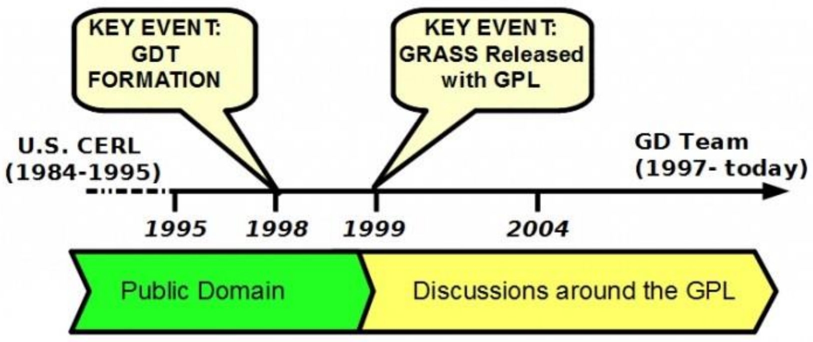

Figure 1 – Key events of the GRASS case study

The license change can be seen as a moment of breakdown. Indeed, GRASS developers were forced to solve Copyright conflicts between the GPL and the licenses of several GRASS commands / modules / libraries. It is in the resolution of these licenses conflicts that we observe the emergence of the “commercial” definition of FLOSS by software developers.

2.1 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 [12] and open. Common examples of open data formats are the Adobe PDF® or the OpenOffice ODT format.

Geographical Information Systems (GISs) 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 closed data format known as DWG. DWG is the native, and proprietary, format of several CAD packages including the well-known AutoCAD®. For a GIS, managing the DWG format is an important feature as many maps have 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 of 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 at the beginning of 2003 as a way to enable 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 by a developer on the GRASS Developers Mailing List:

> 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 OpenDWG library as proprietary software and as a consequence describes the GRASS command *v.in.dwg* as having “a severe license problem”. This is a moment of breakdown: the use of *v.in.dwg*, which has been taken for granted up to that time, becomes problematic as it uses both the OpenDWG library and the GRASS library. Through compilation these two software were becoming a single software and hence a single derivative work of art. In the above message, the developer points out that this is a violation of the GPL license: in brief, for him GRASS could not be distributed in compiled form (i.e. as object code) with the v.in.dwg command. As a consequence, the command v.in.dwg was eliminated from GRASS and therefore the system lacked therefore any possibility to manipulate maps in the DWG data format. Clearly this introduced a serious limitation on the use of GRASS in comparison with, for example, proprietary GISs.

The debate/discussion around the inclusion of the OpenDWG library in GRASS – with hence the possibility for GRASS users to use DWG maps – did not end here. Almost one year after the elimination of OpenDWG library (mid 2003), 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 derivate software that are “free of charge”. For the developer this is a situation that “fits GRASS”. In order to confirm his claims, the developer called the OpenDWG Alliance Head Quarter, asking for clarifications on the use of the OpenDWG library with GRASS.

> 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]

The phone call with the Alliance seems to clarify the doubts about the possibility of using the OpenDWG with GRASS: for the developer the Open Design Alliance license scheme allows the use of the library in non-commercial software and in Free Software.

According to the developer then, it seems possible to distribute GRASS and the OpenDWG library as compiled software as long as GRASS is not commercially sold. From the developer point of view this situation seems 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 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 proposes a definition of *FLOSS as commercial software* and a clear opposition to the antonym between FLOSS and commercial software. Indeed, according to this GRASS developer the GPL requires that the software covered by different licenses needs to 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 [13]. In other words, for this developer 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. This definition (FLOSS as commercial software), states the developer, is also supported by concrete examples such as 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 assumed by the Open Alliance Membership. Therefore the employment of the GRASS command *v.in.dwg* by a user on a local computer is not a direct violation of licenses, however, the use of the same command is a violation of both the GPL and the OpenDWG license for a company that distributes GRASS.

We can easily understand that what is at stake in this discussion is exactly the definition of what is a commercial software and whether GRASS and FLOSS in general can be considered commercial software or not.
In order to clarify this confusion of terms between FLOSS, commercial software 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 [14]

This post clarifies to GRASS developers that it is not possible for them to use the OpenDWG Library together with GRASS, due to the clear commercial nature of GRASS 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 *between FLOSS and non-commercial software* and not between FLOSS and commercial software.

We can see here that the idea of FLOSS being not merely the opposite of commercial software is part of a clear and well-defined strategy of developers: it is an ethnomethod that developers use to make sense and order in their world. When academics oppose FLOSS to commercial software, they impose their terminology on something that the FLOSS actors would disagree on. Several others examples could be taken from the GRASS case to further justify this statement.

At this point, we propose another shorter and revealing example that clarifies that FLOSS developers believe that the antonym of FLOSS is not commercial software but rather proprietary software. These examples relate to the role of the Open Source GeoSpatial foundation (OSGeo) [15] 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 a clarification of the commercial nature of FLOSS is for FLOSS developers a matter of defining what they themselves are and do, it is a matter of identity for FLOSS communities. Indeed, among the developers there is a perceived need of clarification about the precise meaning of the term “commercial”: according to OSGeo spoke-persons 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 as proprietary software. This ethnomethod is very much different from the depicted opposition between FLOSS and commercial software in academic literature.

3. COMMUNITY BUILDING AND COMMERCIAL SOFTWARE: THE CASE OF OPENSOLARIS

The GRASS example shows that the use of the terminology commercial software is part of everyday development practices in voluntary based FLOSS projects, and it is a way 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 between 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 that sprang in 2005 from the release of the proprietary Solaris Operating System with a FLOSS license by Sun Microsystems [16], at the time one of the major global IT player. Later in 2010 Sun was acquired by the Oracle Corporation and since then the project OpenSolaris has been undergoing a series of deep changes, e.g. a fork [17]. Nonetheless, OpenSolaris constituted a major experiment of migration from proprietary software to FLOSS both in technological and organizational terms. Indeed, the 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 almost from scratch a FLOSS community around the system.

Two aspects of the OpenSolaris case are crucial for our discussion: (1) an OpenSolaris-based distribution that Sun provided to its customers; (2) the conversion of the pre-existing proprietary and closed software development into a FLOSS community-based software development. Both these two aspects show an intersection between the commercial nature of Sun and FLOSS. For instance, the definition of OpenSolaris as a commercial project is clearly stated in the first version of the “OpenSolaris Governance Proposal” [18] (later on called “Constitution” [19]) a document that details the governance mechanisms of the project. This document, in its very first sentence, defines 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). In this case, the definition of “commercial-grade operating systems”, which was later discharged, is used to identify the quality of Sun engineering technology as consolidated in the Solaris Operating System, as shown in 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)”. It is clear that in this instance the use of the word “commercial” differs in from the one outlined by GRASS.

A key aspect of the OpenSolaris enterprise can be identified in the Sun Microsystems’ licensing strategy and in the licensing rhetoric used by Sun in relation to the software industry business. Such rhetoric is presented in a book written by two Sun engineers, Goldman and Gabriel (2005, p. 1), who state 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”. Sun engineers believe that FLOSS is a fundamental way of conducting business in the contemporary software market where Innovation Happens Elsewhere, according to the title of the book. In other words, business is seen as a way to harness innovations being developed by others, outside the organization. Hence, the authors articulate 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. This creates fundamental connections between a specific aim (the business) and a set of artefacts designed to increase the participation and 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 are: (1) the software licenses used by Sun and (2) the infrastructure supporting the development. Here we will focus on the OpenSolaris license.

As in GRASS case, the Open Solaris license is a key artefact for understanding the commercial dimension of the system. 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 outlined by Sun taking inspiration from another Open Source License, the Mozilla Public License (MPL). The aim of the CDDL was to build around OpenSolaris a network of entities 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 this type of license is substantially different from program-based licenses such as the GPL. The specific goal of this choice is stated in the first CDDL Frequently Asked Questions [20] (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 as two complementary aspects of Sun licensing and community building strategy, and are clearly not in opposition. It does not matter whether commercial purposes means also distributing derivative works only in binary code: this is still FLOSS, perhaps closer to the Open Source definition than to the Free Software principles pushed by the GPL license. Again, this use of the word "commercial" in relation to FLOSS is very different from what is often presented in academic literature: here commercial software and FLOSS are used together. One of the aims of Sun was to allow the mixing and subsequent distribution of OpenSolaris code files, still protected by the CDDL, with proprietary code. The most notable example is 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 [21].
The modifications that Sun introduced to adapt the MPL license to the creation of the CDDL are an important aspect for understanding the commercial nature of the project. One of the terms of the MPL removed by Sun was the 8.2(b). The reason why Sun decided to remove this term is explained as follows (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*.

In the above statement "any product", according to the original MPL term, meant not only software code but also and in particular any "hardware, or device" covered by a patent [22]. This term is considered an obstacle in enrolling commercial entities especially because of the limitations imposed on hardware. The crucial point of the previous Sun comment to the MPL 8.2(b) term relates again to the community building effort and to the effort of enrolling commercial contributors as well: actors interested in using OpenSolaris in their business activities or, 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 commercially both the source and the binary code of OpenSolaris, and does not just regard the binaries as in proprietary distributions. The answer to the FAQ makes clear that the commercial dimension of OpenSolaris is clearly not referred to proprietary software. Any commercial distribution of binary code derived from CDDL code must always include the distribution of the source code of the released binaries to comply with the copyleft provisions of the CDDL. This makes extremely clear the fact that the use of the term “commercial products” in this instance does not necessarily mean proprietary software distributed only in binary form. In cases in which CDDL code is used together with proprietary software, however, the owner is not required to release the code of its own proprietary software. This point is again made clear in one of the CDDL FAQ, which implicitly states that it is the term “proprietary software” that stands 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 [23], 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. Their use of the word “commercial” is different from the usage of GRASS developers who oppose themselves to the use of any proprietary software. Nonetheless, we also see that FLOSS is not the antonym of “commercial software”, but rather that the two are part of the same project the “commercial Open Source”. Such statement is close to the Open Source Definition.

From the documents we have analysed and presented so far it is clear that OpenSolaris is a commercial initiative that 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, in an attempt to delimit the boundaries of community building through the enhancement of the commercial nature and interest of the project and of those enrolled in it.

It is important to disclose, however, that the constituent commercial nature in OpenSolaris is less clear than in GRASS. This is quite surprising and suggests that the commercial nature of FLOSS needs to be defended more heavily in volunteer-based FLOSS projects. Indeed, in OpenSolaris Mailing list discussions the terminology commercial software is often portrayed by participants as an antonym of FLOSS. This is something that rarely happens in the case of GRASS. For instance, Goldman and Gabriel (2005), the Sun engineers mentioned at the beginning of this paragraph, use 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)

This consideration do not undermine the original thesis of this paper, but show however that reducing the boundary between the term “commercial software” and FLOSS is more crucial for developers in volunteer-based FLOSS projects than in corporate FLOSS projects. In the latter cases, the commercial aspect is taken for granted as a starting point for subsequent practices of detailed definition and legitimization.

4. DISCUSSION AND CONCLUSION

We began this paper by stating how in mainstream FLOSS literature there often is an opposition between FLOSS and the term commercial software. It is important to remark that 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 term "commercial software" as an antonym of FLOSS does not mirror what is happening in the empirical field. In fact, with our empirical, qualitative and bottom-up analysis of the GRASS and OpenSolaris cases we show that this opposition between FLOSS and commercial software is not grounded in current FLOSS development practices and discourses.

We use FLOSS stakeholders ethnomethods to show that the opposition between FLOSS and commercial software outline in academic literature is in contradiction with the empirical world. For instance, our analysis shows 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 allows the free distribution of copies of the software. According to the developers, therefore, GRASS is a commercial software precisely because it is covered by the GPL. 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, including distribution of software in binary code only, as well as facilitating the enrolment of commercial innovators in the community.

A question at this point arises: why academic literature opposes FLOSS to "commercial software" (with its different shades of meanings), 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 argues 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, further lead him to consider that we should speak not just about reality, in singular terms, but about realities in plural terms.

Dick's position on realities helps us emphasize the role of words in the construction of realities. Indeed, what Dick seems to argue is that the manipulation and control of words is not a neutral process, something that does not influence reality. Words are not mere elements that simply represent the things that compose reality rather 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 can be related to the ability of manufacturing a certain reality.
This brief digression into Dick's view of reality helps us to better frame 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 antonyms. What we describe in this paper is the process through which the use of the term *commercial software* in relation to 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 term commercial software from the control of specific discursive practices. This is because the term "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 we observe 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 term “commercial” software in order to pursue its community building strategy. By pushing a specific definition of community building as a commercial enterprise, Sun goals were to enroll commercial innovators into the OpenSolaris community. A clear outcome of both case studies is therefore that controlling the meaning of the term commercial software is for developers a process of manufacturing FLOSS reality.
At this point there is an even more important issue that we need to emphasise: the role of academic discourse. 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. As we show in our Table 1, several important academic publications fully contribute to a definition of reality where FLOSS is sharply opposed to commercial software and where proprietary software is often portrayed as synonym of commercial software. It does not really matter if authors have a nuanced vision of this opposition or whether they better articulate this opposition to show the different shades of meaning of the term “commercial software”. The opposition still portraits a world in which FLOSS is opposed to commercial software, whereas the empirical field presents a world in which FLOSS and commercial software are instead part of the same process.

WORKS CITED

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

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

Bonaccorsi, A. and Rossi, C. 2003. ‘Why Open Source software can succeed’ Research Policy, 32(7): 1243-1258. At http://linkinghub.elsevier.com/retrieve/pii/S0048733303000519. Accessed 10 June 2009.

Callon, M. 1986. ‘Some elements of a sociology of translation: domestication of the scallops and the fishermen of Saint Brieuc Bay’, in: Law J. (ed.) Power, Action and Belief: A New Sociology of Knowledge?, pp. 196-229. London: Routledge.

Chengalur-Smith, I., Sidorova, A., and Daniel, S.L. 2010. ‘Sustainability of Free/Libre Open Source Projects: A Longitudinal Study’, Journal of the Association for Information Systems, 11(11), Article 5. http://aisel.aisnet.org/jais/vol11/iss11/5. Accessed 10 January 2011.

Coleman, G. E. 2005. The Social Construction of Freedom in Free and Open Source Software: Hackers, Ethics, and the Liberal Tradition. Ph.D. Thesis, University of Chicago.

Cornford, T., Shaikh, M. and Ciborra, C. 2010. ‘Hierarchy, Laboratory and Collective: Unveiling Linux as Innovation, Machination and Constitution’, Journal of the Association for Information Systems, 11(12), Article 4. At http://aisel.aisnet.org/jais/vol11/iss12/4. Accessed 27 December 2010.

De Paoli, S. and D’Andrea, V., 2008. ‘How artefacts rule web based communities: practices of Free Software Development’, Int. J. Web Based Communities, 4(2): 199–219.

De Paoli, S., Teli, M. and D’Andrea, V., 2008. ‘Free and open source licenses in community life: Two empirical cases’, FirstMonday, 13(10). At http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/2064. Accessed 10 June 2009.

Dick, P. 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.

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

Glaser, B. G. and Strauss, A.L., 1967. The discovery of grounded theory, Chicago: Aldine.

Goldman, R. and Gabriel, R. 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, A. 2009. ‘Nexenta, Open Storage and Commercial Open Source,’ Presentation at OpenSolaris Developer Conference 28-30 October, 2009, Dresden, Germany. At http://www.osdevcon.org/2009/slides/nexenta_openstorage_anil_gulecha.pdf. Accessed 10 January 2011.

Koch, S. and Schneider, G. 2002. ‘Effort, co-operation and co-ordination in an open source software project: GNOME’, Information Systems Journal, 12(1): 27-42. At http://www3.interscience.wiley.com/cgi-bin/fulltext/118925737/HTMLSTART. Accessed 10 June 2009.

Ke, W. and Zhang, P. 2010. ‘The Effects of Extrinsic Motivations and Satisfaction in Open Source Software Development’, Journal of the Association for Information Systems, 11(12), Article 5. At http://aisel.aisnet.org/jais/vol11/iss12/5. Accessed 27 December 2010.

Kelty, C. 2011. ‘Inventing Copyleft’, in: Biagioli, M., Jaszi, P. and Woodmansee, M. (eds.), Making and Unmaking Intellectual Property: Creative Production in Legal and Cultural Perspective, Chicago, IL: The University of Chicago Press

Kogut B. M. and Metiu, A., 2001. ‘Open Source Software Development and Distributed Innovation’, Oxford Review of Economic Policy, 17(2): 248-264.

Healy, K. and Schussman, A. 2003. ‘The Ecology of Open-Source Software Development’. At http://opensource.mit.edu/papers/healyschussman.pdf. Accessed 10 June 2009.

Hertel, G., Niedner, S. and Herrmann, S. 2003. ‘Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel’, Research Policy, 32(7): 1159-1177.

Lanzara, G. F. and Morner M. 2005. ‘Artifacts rule! How organizing happens in open source software projects’, in: Czarniawska B. and Hernes T. (eds.) Actor-Network Theory and Organizing, pp. 67-90, Liber and Copenhagen Business School Press.

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

Latour B. 2005. Reassembling the Social. An Introduction to Actor-Network-Theory, New York: Oxford University Press.
Law J., 1987. ‘Technology and heterogeneous engineering: The case of Portuguese expansion’, in: Bijker, W. E., Hughes, T.P. and Pinch, T.J. (eds.) The social construction of technological systems: New directions in the sociology and history of technology, pp. 111-134, Cambridge Mass: MIT Press.

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

Lerner J. and J. Tirole 2005. ‘The Scope of Open Source Licensing,’ Journal of Law, Economics, and Organization, 21: 20-56.

Lin Y. 2005. ‘The future of sociology of FLOSS’, FirstMonday, Special Issue #2: Open Source, October 2005. 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’, in: Turner S.P. and Risjord M.W. (eds.) Philosophy of anthropology and sociology, pp. 485-516, 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. 11(3): 309-346.

O’Mahony, S. 2003. Guarding the commons: How community managed software projects protect their work, Research Policy 32: 1179–1198.
Open Source Initiative. 2006. ‘The Open Source Definition’. At http://www.opensource.org/docs/osd. Accessed 28 November 2011.

Perens, B. 2005. ‘The Emerging Economic Paradigm of Open Source’ FirstMonday Special Issue 2. At http://www.firstmonday.org/issues/special10_10/perens/index.html. Accessed 10 June 2009.

Raymond E. S. 1999. 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: Hahn R. W. (ed.), Government Policy toward Open Source Software, pp. 69-85, Washington, DC: Brookings Institution Press.

Sojer, M. and Henkel, J. 2010. ‘Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments’, Journal of the Association for Information Systems, 11(12), Article 2. At http://aisel.aisnet.org/jais/vol11/iss12/2. Accessed 27 December 2010.

Teli, M., Pisanu F. and Hakken D. 2007. ‘The Internet as a Library-of-People: For a Cyberethnography of Online Groups’, Forum Qualitative Sozialforschung / Forum: Qualitative Social Research, 8(3), Art. 33. At http://nbn-resolving.de/urn:nbn:de:0114-fqs0703338. Accessed 10 January 2009.

Tuomi, I. 2000. ‘Internet, innovation, and open source: Actors in the network’, FirstMonday 6(1). At 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, E., and Von Krogh, G. 2003a. ‘Special Issue on Open Source Software Development,’ Research Policy, 32(7): 1149-1157. At http://linkinghub.elsevier.com/retrieve/pii/S0048733303000544. Accessed 10 June 2009.

Von Hippel, E. and Von Krogh G. 2003b. ‘Open source software and the’ private-collective’ innovation model: Issues for organization science’, Organization Science, 14(2): 209-223.

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

Westervelt, J. 2004. ‘GRASS roots. In Proceedings of the FOSS/GRASS users conference’, Bangkok, Thailand, 12–14 September, 2004.

Wheeler, D. A. 2006. ‘Commercial’ is not the opposite of Free-Libre/Open Source Software (FLOSS)’. At http://www.dwheeler.com/essays/commercial-floss.html. Accessed 23 March 2009.

Winograd T. and Flores F. 1986. Understanding Computer and Cognition, Norwood, NJ: Ablex.

Authors short bios:

Stefano De Paoli – Is Research Fellow at the ahref Foundation 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@ahref.eu; Stefano.depaoli@gmail.com

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

Maurizio Teli – PhD in Sociology and Social Research, University of Trento (Italy), is Research Fellow at the ahref Foundation in Trento (Italy). He is involved in and researches about the importance of “practices of freedom” in the processes of organizing online groups and producing technology and narratives. maurizio@maurizioteli.eu

[1] The software known as Freeware is proprietary software distributed gratis, usually for a limited period of time (trial period).
[2] Microsoft Windows, Adobe Acrobat Reader, Solaris, and OpenSolaris are trademarks or registered trademarks of their respective owners.
[3] http://aisel.aisnet.org/jais/vol11/iss12/
[4] http://www.osgeo.org/mailman/listinfo/grass-user

[5] http://www.osgeo.org/mailman/listinfo/grass-dev
[6] http://mail.opensolaris.org/mailman/listinfo/opensolaris-discuss
[7] http://mail.opensolaris.org/pipermail/cab-discuss
[8] http://mail.opensolaris.org/mailman/listinfo/ogb-discuss
[9] http://mail.opensolaris.org/mailman/listinfo/opensolaris-code
[0] http://mail.opensolaris.org/mailman/listinfo/opengrok-discuss
[1] 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.
[2] 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.
[3] In this case the GRASS developers are discussing about 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. "
[4] See http://www.opendesign.com/downloads/guest.htm
[5] 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.
[6] 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
[7] Project Illumos (http://www.illumos.org/), a fork of the OpenSolaris project was launched on August, 3rd 2010.
[8] Retrieved from http://mail.opensolaris.org/pipermail/cab-discuss/2005-July/000763.html
[9] Retrieved from http://www.opensolaris.org/os/community/ogb/governance/
[20] CDDL FAQ, Retrieved at http://openmediacommons.org/CDDL_FAQs.html Accessed 10 January 2010
[2] http://www.sun.com/software/solaris/licensing/sla.xml
[22] See the MPL text: http://www.mozilla.org/MPL/MPL-1.1.html
[23] http://www.nexenta.org/