Computers, Environment and Urban Systems 35 (2011) 173–182

Contents lists available at ScienceDirect

Computers, Environment and Urban Systems

journal homepage: www.elsevier.com/locate/compenvurbsys

Relationality in geoIT software development: How data structures and organization perform together

S. De Paoli [a,*], G. Miscione [b]

[a] Sociology Department and National Institute for Regional and Spatial Analysis, National University of Ireland Maynooth, Iontas Building, Maynooth, Ireland
[b] Faculty of Geo-Information Science and Earth Observation (ITC), University of Twente, Hengelosestraat 99, PO Box 6, 7500 AA Enschede, The Netherlands

ARTICLE INFO

Article history:
Received 4 March 2010
Received in revised form 13 September 2010
Accepted 15 September 2010

Keywords:
Constructivism
Relationality
GIS
Data structure
Memory allocation
Organization

ABSTRACT

Constructivism in geo-information science has emphasized what happens to geo-information technologies (geoIT) after the design stage, when systems and applications are used in real life. Current constructivist views, however, have focused less on other aspects such as software development practices. This paper adopts a similar constructivist epistemology, but looks at how geoIT and people are entangled in the development stages.

We discuss the case of the migration of GIS software to Free and Open Source license. This case provides clear empirical evidence of the entanglement of humans and artifacts during the development of GIS technologies. Through an analysis of archived material (such as mailing lists), and of the software code, the paper describes how the integration of a new software (the library Fast Fourier Transformation in the West) was hindered by the different data structures of the original GIS and the new software. The case study we propose shows how actual software development practices may contrast with the well-established rhetoric of technical efficiency of the algorithms. In addition this choice also illustrates the organizational aspects of developing GIS and the different weights that are given to computational resources and organizational resources.

1. Introduction

Constructivism in geo-information technologies (geoIT) is an approach that has criticized the limits of positivist approaches (e.g. Goodchild, 1998) and the idea of a one-way impact of geoIT on society (e.g. Pickles, 1995). Constructivism therefore rejects positivist and techno-deterministic approaches, emphasizing on the contrary the mutual constitution of technology and society. This is what Chrisman (2005) calls the *Full Circle*, in which software shapes social relationships and is shaped by them:

> Certainly there are impacts of the new technology on society, but these can only be understood when we also consider the impacts of society on the technology. By turning full circle, connecting from social needs to technical issues, then back to the social realm, we avoid the flaws of isolating implications from their causative environment. (Chrisman, 2005, p. 32)

Authors such as Chrisman and Harvey (Chrisman, 2005, 2006; Chrisman & Harvey, 1998, 2004; Harvey, 2000, 2009) and Schuurman (Schuurman, 2002; Schuurman & Pratt, 2002) have promoted an approach to geoIT that is based on constructivist concepts mainly drawn from Science and Technology Studies, for example using the concepts of 'boundary object' (Star & Griesemer, 1989) or 'configuring the user' (Woolgar, 1991). Constructivism has the merit of emphasizing that technical choices inside geoIT and geographic information systems' (GIS) software have a fundamental social dimension, and this constitutes an antidote to determinism (Chrisman, 2005).

In this paper, while we agree with the idea that geoIT possesses a crucial social dimension, we also consider that current constructivist literature has overlooked what happens to geoIT during software development stages, before applications, systems or geographic data formats will be used in real life settings. We aim to avoid both technological and social determinism by discussing a long term software development process, and showing how geoIT and organizational processes perform together. Therefore, following a constructivist approach, our goal is to investigate some aspects of the development stages of geoIT with the goal of unveiling the mutual shaping of technologies and organizational aspects (the *Full Circle*) that happens during these stages. We are of the idea that our view complements already existing constructivist analyses. Indeed we fill a gap in the literature by proposing empirically based knowledge on what happens before technologies are adopted by end users.

GeoITs are composed of various technologies that include hardware and software. In this paper, we focus on the latter,

* Corresponding author. Tel.: +353 17086688.
E-mail addresses: stefano.depaoli@nuim.ie, Stefano.depaoli@gmail.com (S. De Paoli), g.miscione@utwente.nl (G. Miscione).

0198-9715/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compenvurbsys.2010.09.003

particularly on the co-evolutionary relationship of GIS software and its developers, as a case of *entanglement of humans and artifacts* (Orlikowski & Scott, 2008) and of imbrication of geography and technology (Chrisman & Harvey, 2004). Criticizing mainstream organization studies on information systems, Orlikowski and Scott argue that the social and the technical are not separate entities, but that they "perform" together. Our investigation on the entanglement between GIS software and its developers relies upon the results of an empirical case study of the migration of Geographic Resources Analysis Support System (GRASS, see Neteler & Mitasova, 2002) under a Free and Open Source Software (FOSS) license, the GNU General Public License V. 2.0 (GPL hereafter).[1] The events we refer to in this paper happened mostly during the years 1999 and 2001.

What is interesting about the GRASS case study is that the migration to the GPL license forced the GRASS developers into a series of socio-technical negotiations. In particular, the specific and declared goal of these negotiations was to substitute several software modules/programs of GRASS (whose licenses were legally incompatible with the new license) with other software performing the same operations but compatible with the GPL. From our point of view, this migration was an occasion to follow GIS software development practices and problems-solving. Thus, FOSS is not our research focus per se, but the GRASS case study proved to be an opportunity to unpack the black-box (Bijker, Huges, & Pinch, 1987) of GIS software development and to observe underlying socio-technical logic and technological efficiency construction.

In this paper, we analyze a specific case of software substitution, that of the code for computing the Fast Fourier Transformation (FFT hereafter), an algorithm that computes the Discrete Fourier Transformation. In GIS, the FFT is used for processing and filtering remote sensing images. Indeed, with the adoption of the GPL, the GRASS development team undertook a process for the migration from the Numerical Recipes' (Press, Teukolsky, Vetterling, & Flannery, 1988) implementation of the FFT to the FOSS library FFTW.[2] By describing the socio-technical practices undertaken by the GRASS development team to operate this substitution, we show the entanglement between the organizational constraints (mainly in terms of resources available) faced by the GRASS development team for the development and the final shape of the technical solution.

The paper is organized as follows: first we present our theoretical framework and our take on constructivism; consequently, we describe our research design and methods. In Section 4, after introducing GRASS as an empirical case, we explain why the license change provided a good opportunity for research to gain insights into the black-box of GIS software development. We then move towards the case study and describe the process of substitution of the Fast Fourier Transformation software. In Section 6 we propose reflections on the social and technical relationality in geoIT and reflect on how technological efficiency was shown to be an outcome rather than a principle of GIS software development.

2. Socio-material relationality

The theoretical starting point of this paper is the following: we consider that current constructivist contributions to literature focus more on the socially constructed nature of geoIT use, and less on other aspects of the social construction of geoIT such as software development. As support to this consideration, we can for instance observe that the recent *Journal of Information Society* special issue on geoIT (2009) did not contain papers about the actual production of geoIT. The editors highlight the vitality of research on social aspects of geoIT in the last decade:

> researchers from geography and neighboring disciplines have since tackled many key and critical issues, specifically around the three initiatives of the societal component of the Varenius Project: (i) place and identity in an age of technologically regulated movement, (ii) measuring and representing accessibility in the information age, and (iii) empowerment, marginalization, and public participation geographic information systems (PPGIS). (Ekbia & Schuurman, 2009)

This special issue includes contributions along three main dimensions: geographical, social, and informational. They manifest themselves in contributions on land use and decision making (Harvey, 2009), religious communities' use of online environments (Cheong, Poon, Huang, & Casas, 2009), mobile communication and urban mobility (Kim, 2009), wireless networks and construction of places (Forlano, 2009), and Spatial Data Infrastructure and its travel across space and time (Homburg & Georgiadou, 2009). No article of the special issue considers, however, the social dimension of geoIT construction in terms of software development, an aspect of geoIT that remains therefore black-boxed.

An initial answer to the limits of social constructivism in un-packing geoIT development comes from two works by Chrisman. In a relatively recent book, Chrisman (2006) describes how GIS mapping technologies came out of Harvard in the period of the 1960s onward. Although this book is mostly a historical account of events, Chrisman also discusses how negotiations among a variety of actors (researchers, institutions) and technologies (available mapping techniques and computing resources) lead to GIS as we know it today.

In a second work, Chrisman (2001) follows Woolgar (1991) illustrating how GIS technological choices can configure different users, hence giving birth to different divisions of labor within the field of GIS practice. For example, Chrisman describes the case of how different geographic representations based either on separated layers (e.g. polygon overlay) or on a gestalt view of the map (integrated terrain map view) were adopted by different communities of practitioners in the field (public administration in the first case, academia and public agencies in the second). According to Chrisman neither of these mapping techniques can be said to be the most accurate. However, according to the author, the preference given to the overlay technique in public administration reflects "administrative hierarchy, with its implicit divisions of labor and responsibility" (p. 10), whereas the preference given to gestalt mapping by public agencies reflects their autonomy.

A further example comes from D'Andrea and De Paoli (2008), in which the authors show that the final configuration of a GIS technology and its users depends also on legal issues, such as copyright licenses compatibility among GIS software components.

The two works by Chrisman (2001, 2006) and D'Andrea and De Paoli (2008) are not entirely concerned with software development as such, but with geoIT technologies in the broad sense (e.g. geographic representation, computational resources and research funding or legal issues). Therefore they constitute a basis for our constructivist view in investigating the relations and co-construction between technological development and organizations.

With this paper our aim is to fill the gap in constructivist literature – the focus on geoIT use and the rare attention given to software production – by studying the development of software. By doing so, we also aim at bridging a tacit division of labor between studies on technological and on social aspects of geoIT and to contributing to a better understanding of the artifacts' co-construction. Indeed, we aim at un-packing the social dimension of GIS source code development to add to a major theme of constructivism approaches (Bijker, 1995; Bijker et al., 1987): that the

[1] The GPL license is the main Free and Open Software license used on thousands of software programs including, among others, the well known operating system Linux.

[2] FFTW stand for Fast Fourier Transformation in the West (http://www.fftw.org). FFTW is a C subroutine library for computing the discrete Fourier transform (DFT).

efficiency of a technology is not what explains its success, rather the efficiency (and inefficiency) of technologies need to be explained as socially (or socio-technically) constructed. We argue that the efficiency of geoIT is an effect rather than a cause of technological and organizational drivers. Orlikowski (2000; Orlikowski and Iacono, 2001) places the traditional dichotomy between structures and agencies at the analytical (rather than empirical) level. Adopting such a stance, the author emphasizes how artifacts reproduce structures through actions and, symmetrically, emphasize if and how structures enact artifacts. Structuration theory rejects unilateral views based on the impact of technology on organizations (as auditing, for instance) or the impact of organizations on technology (as explicit "users' requirements"). Structuration theory allows studying how technology enacts structures, and how actions enable technology reciprocally. In this line of reasoning, Orlikowski and Barley (2001) strengthen the view on the mutual interdependence of technology and organization by arguing that Information Systems research and Organizational Studies can learn from each other: the former by including institutional analysis, the latter by considering the specific characteristics of IT.

In a recent work, closer to our theoretical perspective, Orlikowski and Scott (2008, p.21) look at the strand of research on "socio-materiality". The authors argue that the socio-technical paradigm conceives the social and the technical as self contained entities interacting with each other, whereas a socio-material perspective understands them as they exist in a mutual relation. Therefore, for the authors, the key mechanism that produces action is not the "interaction" between human and non-human (Latour, 1987) but rather "performativity". In other words, for Orlikowski and Scott, the social and technical perform the action together, so they are "entangled". The authors also refer to Suchman (2007, p.268) who argued that socio-materiality accepts the mutual constitution of actors, but not in the same way. In conclusion, in this paper we adopt a constructivist epistemology well represented by Orlikowski and Scott's work, and look at how geoIT and organizations are entangled in the development process of GIS software. The technical choices from the empirical case will be discussed by highlighting how software (in)efficiency is "performed" by a variety of social and technical actors (what in ANT terms are defined as actants or entities): the technicalities of remote sensing, conditions pre-existing the FOSS switch, the development team's organizational form and actual resources.

3. Research approach

To be coherent with our theoretical framework, we adopt a methodological principle whereby the researcher does not decide in advance the social and technical attributes of the technological system. Therefore, we do not decide in advance the performativity of technology and society, but we seek to explain how these perform together. This also implies that the observer is required not to impose in advance a theory to understand the socio-technical change. Callon (1986) describes this emergent 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 natural." (pp. 200–201)

Methodologically, our investigation of the GRASS case study scrutinizes both software development practices and technologies themselves, that in FOSS, by definition, are mostly carried out in the open (Raymond & Moen, 2006).

The empirical research has been conducted through the analysis of archived material of both online interactions (such as mailing lists), and of the software source code. The empirical data in this paper comes from a 3 year long in-depth investigation of the GRASS case study (during the period 2003–2007), conducted as part of the doctoral research of one of the authors.

GRASS, as many other FOSS projects, is mostly developed through the means of the Internet infrastructure and by geographically dispersed development teams. All the tools that are used in FOSS development (such as mailing lists, shared repositories of source code, web-sites or IRC channels) are important sources of data for empirical investigations. Specifically, the data presented in this paper comes from the investigation of GRASS archives, including the GRASS User Mailing List (1991–2010, GUML hereafter) and the GRASS Developers' Mailing List (1991–2010, GUDML hereafter) archives and source code archives. Some data also refers to the direct participation in the mailing list discussions.

More precisely, we present data that refers to the mailing list public discussions related to the elimination of the functions from the Numerical Recipes implementation of the FFT and its substitution with the FFTW. The analytical approach of this research emphasizes the accounts that are provided directly by the actors themselves (Callon, 1986; Latour, 1988): in particular by GRASS developers and users. Moreover, in the analysis we kept track of how design details and the already existing software solutions were active entities of concrete negotiations, including people's practices and orientations. In other words, we considered the role of what Law (2004, p. 13) defines as the "hinterland of pre-existing social and material realities". In this line, adopting a concept traditionally closer to IT research, we purposefully considered the role of the "installed base" (Hanseth & Monteiro, 1997; Monteiro & Hanseth, 1995) to highlight the variety of shaping socio-technical arrangements. The concept of installed base implies that technical infrastructures always already exist in one form or another, and that the existing elements of an infrastructure influence its future development (Bowker & Star, 1999; Star & Bowker, 2002). Thus, the installed base that includes not only artifacts but human habits, norms, and roles (Edwards, Bowker, Jackson, & Williams, 2009, p. 366), provides both possibilities and constraints for infrastructural evolution. Radical and abrupt changes are indeed rare and intervention attempts need to take into account the inertia or flexibility of the already existing software and organizational resources. New parts and new organizational practices are integrated into an existing installed base through the extension of the latter or the replacement of existing parts. In this way the installed base evolves creating inertia (self-enforcement with the effects of path-dependence, lock-in, and possible inefficiencies) (Hanseth & Monteiro, 1998). In addition, the distributed nature of mandate, ownership and agency creates several obstacles for the applicability of conventional and control-oriented management approaches (Ciborra & Associates, 2000).

4. Brief history of GRASS

GRASS was born at the beginning of the 1980s as a small project of the United States Army Corps of Engineering Research Laboratory (USACERL). The system was distributed by the USACERL as public domain[3] software. 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 (US Army CERL, 1996) announced its decision to stop

[3] Public Domain is a legal term that states that there is no Copyright over a work of art, like software.

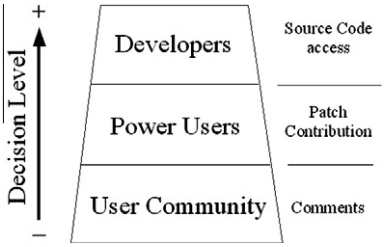

Fig. 1. Organizational structure of the GRASS Development Team, adapted from Neteler (2006).

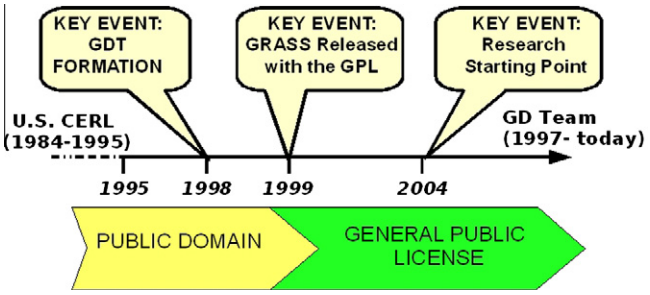

Fig. 2. A synopsis of the GRASS case study.

GRASS development following a governmental decision. 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 continue to develop GRASS as a FOSS. The GDT included (and still includes) a group of volunteers affiliated to several public and private international organizations. The new development team took a structure close to the "town council" model (Cox, 1998), characterized by a small group of programmers leading the development of a large project. This partly differs from the classic idea of a Bazaar development (i.e. a flat organization, with a limited centralized control and a large number of developers), but also differs from the Benevolent Dictator model (i.e. a single person acting as gateway to all the development decisions) (Rivlin, 2003) (see Fig. 1).

The current versions of GRASS (for example GRASS 6.4) are composed of more than eight-hundred thousands lines of source code written in the C programming language (GDT, 2010a). The development is currently led by a worldwide team of developers (about 38 people, of which more than half are active) and sustained by an estimated user base of 25–30,000 users,[4] distributed all around the world.

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 GPL license, version 2 (FSF, 1991). The GPL is well known for a specific licensing term: the "copyleft". The copyleft term states that derivative works based on previous GPL'ed software, must be GPL'ed as well. In this way the license ensures that the source code of the software is not only always available to the public under the same copyright conditions, but can also be modified and distributed as long as the original license's terms remain intact. In other words, the copyleft clause is hereditary and once the license it is applied to a piece of software it remains on that and on its derivative. More relevant for us here, is that GPL made the process of software development transparent to us.

The release under the GPL has made GRASS one of the leading FOSS geoITs world wide. Indeed, GRASS has gained world wide interest and today is one of the founding projects of the Open Source Geospatial Foundation (see Fig. 2).[5]

5. What the license shift made visible

Numerical Recipes (NRs) is the title of an influential series of books on algorithms and numerical analysis (Press et al., 1988). The book series contains a huge amount of material related to computational methods, with examples of implementation of algorithms for numerical analysis in different programming languages. The term Numerical Recipes does not refer just to the book series. In fact each book is always accompanied by the implementation of the algorithms, for example Numerical Recipes in *C*; Numerical Recipes in *Fortran* and so on. Indeed, as is noted on the authors' website:

> "Numerical Recipes" also refers to the copyrighted computer software that is in those books, and also sold separately.[6]

For a developer the possibility to use the NRs in software, is therefore tied up with copyright compliance. As the authors of NRs noticed:

> We receive a range of requests regarding redistribution permissions, and we try to apply a consistent and straightforward policy in answering these requests.[7]

The NRs' authors allow therefore the use of NRs' software under specific circumstances. For example the use of NRs' software is allowed in executable code for non-commercial use only, while the distribution of the software source code of NRs' software (still for non-commercial use) is prohibited.[8]

This short digression helps to frame the empirical problem of this paper: GRASS contained the implementation of some NRs' algorithms. The following message, from the GRASS Developers' Mailing List (GDML), remarks how the inclusion of NRs in GRASS was somehow problematic because of the GPL:

> had you noticed the problematic inclusion of the 'Numerical Recipes' code in the now-GPL'd GRASS v5.0b4?
> [GDML, 14 December 1999, http://lists.osgeo.org/pipermail/grass-dev/1999-December/012746.html]

The use of NRs' software in GRASS dated back to the period when the system was still developed by the US Army. The NRs' software in GRASS was covered by the following copyright notice:

> /* Based on "Numerical Recipes in C; The Art of Scientific Computing"
> (Cambridge University Press, 1988). Copyright (C) 1986, 1988 by Numerical Recipes Software. **Permission is granted for unlimited**
> **use within GRASS only**.*/
> and comes down to three relatively small functions:
> "egvorder" (eigsrt) function
> "fft" (fourn) function
> "jacobi" function
> [GDML, 14 December 1999, http://lists.osgeo.org/pipermail/grass-dev/1999-December/012746.html, *bold emphasis added*]

From this email, which was posted by a GRASS developer on the GDML, we can see that the authors of "Numerical Recipes in C" granted the use of three specific functions to the US Army GRASS developers. The use of these three NRs functions was also granted to the GRASS users. In particular, according to an email sent by the

[4] The number is an estimate based on downloads of the GRASS software.
[5] See OSGeo Portal at http://www.osgeo.org.
[6] From http://www.numerical-recipes.com/infotop.html.
[7] From http://www.numerical-recipes.com/infotop.html.
[8] From http://www.numerical-recipes.com/infotop.html.

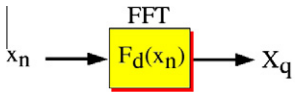

Fig. 3. Schematic representation of the FFT: it creates a relation between two series of numbers (*xn* input and *Xq* output).

NRs' authors to the US Army developers (that was also reported on the GDML), the use of these three functions was granted only for "*a non-commercial research purpose*". In addition for the NRs' authors it was a matter of concern to "*make clear to your users* [the GRASS users] *that they have permission to use the included routines *only* with your software, *not* to transplant it to other programs*".[9] (text in square brackets added).

With the adoption of the GPL as the GRASS License arose the problem of assessing the compatibility between the NRs' copyright and the new GRASS License. According to some GDT members the agreement between the NRs' authors and the US Army was still valid. For other developers there was, however, a clear incompatibility with the GPL: indeed the NRs' copyright notice not only prevents users from further developing the code, but also prevents its inclusion in other software (i.e. permission is granted for unlimited use within GRASS only). This clearly clashes with the "copyleft" clause of the GPL that requires the software to be further modifiable for users. Hence, due to a license incompatibility, the GDT was forced to eliminate the NRs' software from GRASS.

5.1. How to eliminate the NRs by creating a different object

The elimination of the NRs from GRASS involved a series of socio-technical negotiations between the GRASS Development Team and the GRASS system. For example one of the NRs' elements of software contained in GRASS was an implementation of the algorithm known as Fast Fourier Transformation (FFT hereafter). This algorithm is a mathematical method used for the computation of the Discrete Fourier Transformation, whose goal is to transform a mathematical function into its frequency domain. For our discussion, more important than the mathematical problem per se (see Howell, 2001) is to consider that the Fourier Transformation creates a relation between two series of numbers. These series are usually (but not always) composed of complex numbers.

A complex number is a real number to which it is added an imaginary part, designated usually by the *i* character. Hence a complex number *Z* is represented as $Z = a + bi$, where *a* and *b* are real numbers, and *i* is the imaginary unit (Howell, 2001). The real number *a* is called the real part of the complex number, while *b* is called the imaginary part. Hence, given a series of numbers *xn* with $n = [0, 1, \ldots, N-1]$ we can schematically represent the Fourier Transformation as the series of numbers *Xq* with $q = [0, 1, \ldots, N-1]$, as follows (see Fig. 3):

What is important for our discussion is the consideration that the FFT creates a relation between two series of numbers, *xn* as input and *Xq* as output.

In computer science, a common application of the FFT is the manipulation of images. The FFT is indeed used for a broad range of image processing, such as the analysis, filtering, reconstruction and compression of images.[10] For example, in Remote Sensing Techniques raw images require a certain level of manipulation and correction before any interpretation may be possible (Martin, 1996; Schowengerdt, 1997). Among these corrections and manipulations, the Fourier Transformation allows the identification and elimination of periodic noises from images.

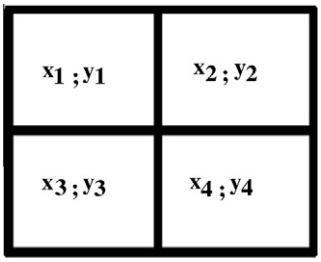

Fig. 4. Raster layer.

5.2. The case of the FFT

Going back to GRASS, the NRs' implementation of the FFT (hereafter FFT-NR) was a function integrated in the mathematical library of the system. This function was used by some GRASS programs/modules to compute the FFT. For example the GRASS module/program i.fft (GDT, 2010b) is an image processing program based on the FFT-NR, that processes a single input raster map layer (input_image) and constructs as output the real and imaginary Fourier components in frequency space (Neteler & Mitasova, 2002, p. 239). Hence each pixel of a *raster layer* acquired through a Satellite, contains a pair of real numbers (*x*; *y*) with specific values. On this *raster layer* it is then possible to compute the Fourier Transformation by using the GRASS command i.fft (see Fig. 4).

The GRASS command i.fft receives as input the raster layer, uses the FFT-NR contained in the GRASS mathematical library and returns as output two *raster layers*: an image composed of the real (*rxn*; *ryn*) and an image composed of the imaginary (*ixn*; *iyn*) Fourier components (see Fig. 5).

5.3. Elimination of FFT-NR

As already noticed, due to the incompatibility between the GPL and the NRs copyright statement contained in GRASS, the GDT was forced to eliminate the FFT-NR implementation from the system. The GDT was however also forced to provide GRASS users with a new solution for the computation of the FFT, a solution fully compatible with the new license. In particular the GDT adopted a GPL'ed library known as the Fast Fourier Transformation in the West (FFTW hereafter). However, this (mandatory) decision to eliminate the FFT-NR and its substitution with the FFTW came with some problems as described in the following email posted on the GRASS User Mailing List (GUML):

> a) adds another library dependency,
>
> b) requires either that existing applications are re-written to use the FFTW interface, or that we add code to convert between the existing and FFTW interfaces (which might introduce inefficiency; I don't know the semantics of the existing interface, so I can't tell).
>
> [GUML, 09 August 2001, http://lists.osgeo.org/pipermail/grass-dev/2001-August/003082.html]

In this message, a member of the GDT highlights some consequences related to the adoption of the FFTW. First, the FFTW will introduce a library dependency. In other words, while the NRs' software was fully incorporated in the GRASS mathematical library, the FFTW will operate "externally". This is a *dependency*, a

[9] From GDML http://lists.osgeo.org/pipermail/grass-dev/1999-December/012745.html.

[10] See http://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm.

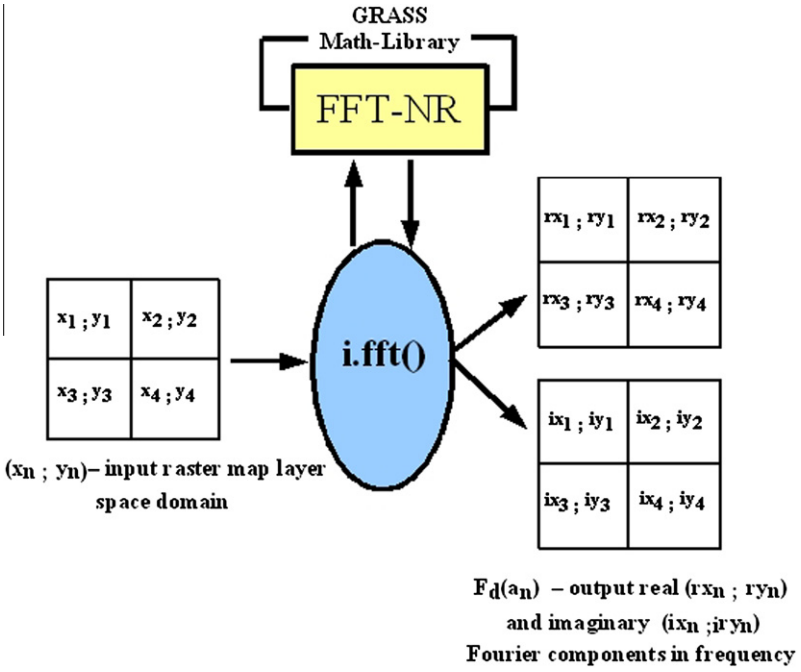

Fig. 5. Computation of the FFT by the GRASS commands i.fft, using the FFT-NR.

situation in which software, in order to be fully functional, depends on the functionalities of other external software. The FFTW dependency is however only a secondary problem here. GRASS users can indeed download the FFTW and install it together with GRASS.

5.3.1. Different data structures

It is the second part of the message that we have to consider more carefully. The developer notices that the GRASS programs/modules using the FFT-NR (such as i.fft) would have required to be partly re-written to adhere with the FFTW data structures. An alternative solution to rewriting the data structures of the GRASS modules was to write a program interface (an FFT Interface) to be placed between the GRASS modules and the FFTW library. Between these two possible options the GDT decided to adopt the second one: they decided to write a simple program FFT Interface that translates between two different data structures. The following email clarifies this:

> The old fft() function (*NRs*) used separate arrays for the real and imaginary components, while FFTW uses an array of fftw_complex values (i.e. interleaved real and imaginary components).
>
> The old fft() function required the array dimensions to be powers of 2, while FFTW doesn't have this requirement.
>
> [GUML, 04 May, 2005, http://lists.osgeo.org/pipermail/grass-dev/2005-May/018304.html]

As already described, the FFT algorithm creates a relation between an array[11] of numbers as input and an array of complex numbers as output. The previous email messages highlight the existence of a fundamental difference between the FFT-NR and the FFTW in manipulating these arrays: the data structures.

In the first case (FFT-NR) the real and imaginary parts of the complex number (both input and output) are stored using two separate arrays. For example, considering only an array X of real numbers as input, we will have as output of the FFT-NR two different arrays, one for the real Fourier part (R) and one for the imaginary part (I) (see Fig. 6):

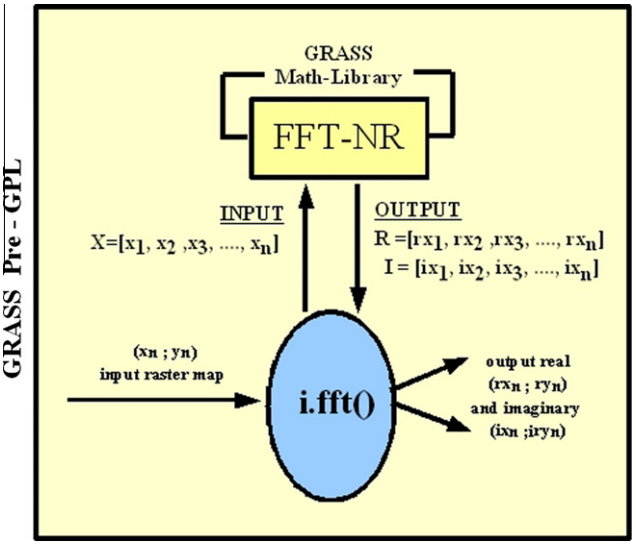

Fig. 6. FFT-NR and GRASS before the GPL.

$$R = [rx1, rx2, rx3, \ldots, rxn] \quad \text{and } I = [ix1, ix2, ix3, \ldots, ixn]$$

In the FFTW the data structure consists instead of a unique array of complex numbers (called fftw_complex). Hence, with X as input, we have as output an array Z of complex numbers (see Fig. 7):

$$Z = [rx1 + ix1, rx2 + ix2, rx3 + ix3, \ldots, rxn + ixn]$$

Hence, between the FFT-NR and FFTW there are two different data structures in which the complex numbers are stored either in two separate arrays (R and I) or in a unique array of complex numbers (Z). The GRASS programs using the FFT-NR, such as the

[11] An *array* is a type of data structure used in many programming languages that allows to define new data types from existing types. An array is a list of records, whose boxes are cells in the array: each cell is a variable of the same pre-existing data type base of the array.

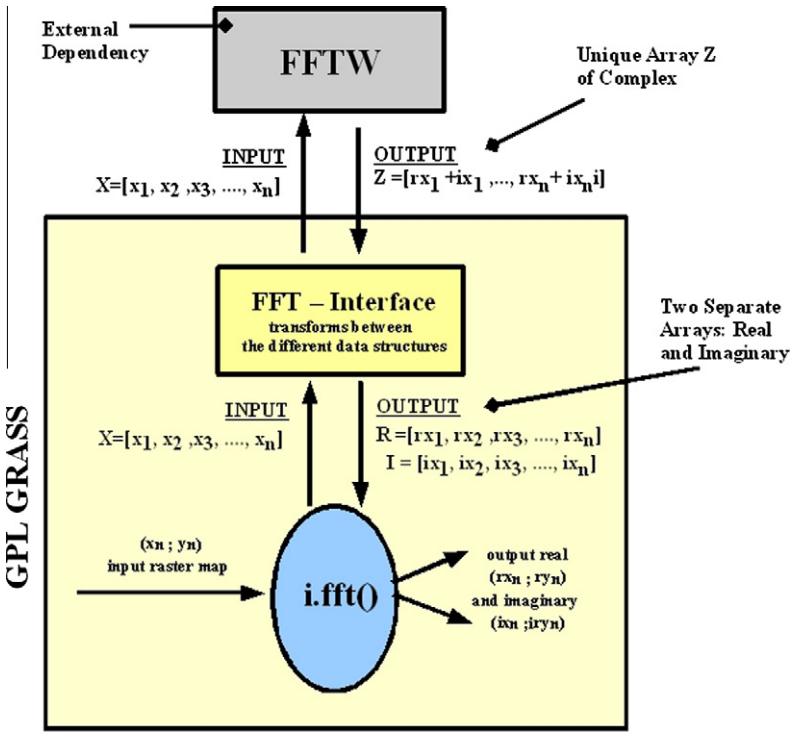

Fig. 7. The introduction of the FFT interface, with the goal to transform between two different data structures.

i.fft, did have the same data structure as the FFT-NR. In other words the real and imaginary parts of the complex numbers were stored by the GRASS programs/modules (such as the i.fft) using two different arrays. This is the reason why the GRASS development team had the problem of either rewriting the data structure of the GRASS modules (to make them adhere with the FFTW data structure: a single array *Z* of complex numbers) or to write a program interface (an FFT Interface) in order to transform between the two different data structures. The GDT decided to adopt the second solution, therefore in order to use the FFTW, as the following email remarks:

> Most (or all) of the existing users of the fft() function are converting data to and from the format which the old fft() function used, and the new fft() function is converting it to or from the format which FFTW wants.
>
> [GUML, 04 May, 2005, http://lists.osgeo.org/pipermail/grass-dev/2005-May/018304.html]

The following two figures (Figs. 6 and 7) try to explain the content of this email message. Before the introduction of the GPL (Fig. 6), there was a total correspondence between the data structures of GRASS modules (e.g. i.fft) and that of the FFT-NR. The following figure shows this adherence by assuming (as simplification) just one array (*X*) as input of the FFT-NR. We have one array *X* as input and two arrays as output, one for the real part (*R*) and one for the imaginary part (*I*). In addition it is important to note that the FFT-NR is fully integrated inside the GRASS Math Library (i.e. it is not a dependency).

The following figure (Fig. 7) instead, shows the situation after the adoption of the GPL with the introduction of the FFTW. First of all the FFTW is external to GRASS: there is therefore a dependency. What is fundamental however is that the GDT added a program "FFT Interface". The goal of the interface is that of transforming between the different data structures: that of the FFTW (a single array of complex numbers) and that of the GRASS commands (two arrays real and imaginary).

5.3.2. The memory allocation problem

The different data structures were not the only fundamental difference between the FFT-NR and the FFTW. A second problem faced by the GDT was the following:

> "The old fft() function required the array dimensions to be powers of 2, while FFTW doesn't have this requirement.".
> [GUML, 04 May, 2005, http://lists.osgeo.org/pipermail/grass-dev/2005-May/018304.html]

According to this message, the FFT-NR requires for its execution arrays whose dimension is the next power of 2. For example with an array of 20 elements the FFT-NR requires an array of 32 cells (2^5), that is the next power of 2 (where $2^4 = 16$). For instance, with a raster map of 256×256 pixels ($2^8 \times 2^8$) as input, the FFT-NR processes the image with a matrix of dimension 256×256. As each pixel of the map is 32 bytes, then the execution of the FFT-NR requires $256 \times 256 \times 32$ bytes of memory. With a map of 200×400 pixels, the FFT-NR requires to approximate to the next power of 2: in this case $256 \times 512 \times 32$ bytes ($2^8 \times 2^9$).

The FFTW does not require any array approximation. The FFTW processes the arrays at same dimension. For example a map of 200×400 pixels is processed at this same dimension, 200×400. However given that the GRASS modules adhere to the FFT-NR they also required an approximation to next power of 2. The introduction of the program "FFT Interface" between i.fft and the FFTW, created therefore a further problem, in relation to some specific situations, as the following message well describes:

> The net result of this is that code which uses fft() is wasting a lot of memory. In the worst case, it can use almost 8 times as much memory as is actually necessary. Padding each dimension to the next power of 2 can result in a near-fourfold increase, while storing two copies (the separate real/imaginary arrays plus the interleaved array) doubles it again.
> [GUML, 04 May, 2005, http://lists.osgeo.org/pipermail/grass-dev/2005-May/018304.html]

This message requires a few clarifications. First the program i.fft, in order to compute the Fourier Transformation of a *raster layer*, must do an operation known as *memory allocation*. The dynamic memory allocation is the allocation of an amount of physical memory for use in a computer program during the execution of that program. The memory allocation is a way of managing and distributing limited memory resources among various programs in execution. In other words with a dynamic allocation the computer program establishes in advance the amount of memory it requires for the execution of the task.

The i.fft program requires a memory allocation for storing the input map. The amount of memory required by the GRASS programs (such as i.fft) for the execution of the FFT, as we have noted before, is an approximation to the next power of 2. Hence, a map of 200×400 pixels requires a memory allocation of $256 \times 512 \times 32$ bytes. For example with a map of 2049×4097 pixels (notice that $2048 = 2^{11}$ and $4096 = 2^{12}$), the i.fft requires an allocation of $4096 \times 8912 \times 32$ bytes ($2^{12} \times 2^{13}$). Therefore, with the requirement of having *arrays* with a dimension which is power of 2, we have the first problem: in the worst case (e.g. 2049×4097 vs. 4096×8912) the programs (e.g. i.fft) require an allocation of almost double the amount of memory.

The second inefficiency relates to the operations of the program "FFT Interface". In order to make the transformation between the two different data structures (that of FFTW and that of for example i.fft), the program "interface" allocates itself a certain amount of memory. In other words the interface must allocate a sufficient quantity of memory to transform the array of complex numbers *Z* (the FFTW data structure) in the two arrays *R* (for real numbers) and *I* (for imaginary) (the FFT-NR data structure) and vice versa (and so the allocation of memory doubles).

Both these inefficiencies (the power of 2 "rule" and the double allocation of memory of the "interface") lead, in the worst case, to a memory allocation eight times bigger than that effectively needed for computing the Fourier Transformation with GRASS. Both these problems could have been solved if the data structure of FFTW and the GRASS modules (e.g. i.fft) were the same.

5.3.3. Users facing the inefficiency

At this point it is interesting to see how the memory allocation inefficiency was in some rare occasions affecting the GRASS use. The following email, coming from the GRASS User Mailing List, is the example (one of the few though) of a user having problems in executing the FFT with GRASS:

> I figured out, that reducing the size of a window will make i.fft work.
>
> My original regions, where it segfaults:
>
> [...]
>
> rows: 11923
>
> cols: 25151
>
> The region where it seems to work well:
>
> [...]
>
> rows: 5445
>
> cols: 6519
>
> [GUML, 10 June 2004, http://lists.osgeo.org/pipermail/grass-dev/2004-June/014825.html]

In this message a user complains about a problem in using the i.fft command: with a *raster layer* of 11923×25151 pixels (*original regions*) as input of the i.fft, the user obtains a "segfault" error.[12] However, with a reduction of the *raster layer* dimension to 5445×6519 pixels, the error disappears. A developer explains the reasons for the segfault error as follows:

> So, you would actually need:
>
> $$16384 * 32768 * 32\,\text{bytes}(4\,\text{doubles}) = 17179869184(16\,\text{Gb})$$
>
> for the original resolution, or:
>
> $$8192 * 8192 * 32\,\text{bytes}(4\text{doubles}) = 2147483648(2\,\text{Gb})$$
>
> for the reduced resolution.
> [...]
> The memory requirements could be reduced to two doubles per cell (without scaling up to a power of two) if i.fft used FFTW directly, rather than using the fft() interface.
> [GUML, 10 June 2004, http://lists.osgeo.org/pipermail/grass-dev/2004-June/014830.html]

In other words, with big maps the program interface that transforms between the two data structures might require more memory than that effectively available on the user's computer: in the first case (with 16,384 rows and 32,768 columns) it requires an allocation of 16 GB of RAM, more than the memory normally available on the computer at the time of these messages. As a result the user cannot compute the Fourier Transformation on the original map.

5.4. Was this an inefficiency of GRASS?

At this point of the paper we have described in full details the negotiations undertaken by GRASS Development Team in order to eliminate the FFT-NR from GRASS. We have in particular highlighted how the solution adopted by the GDT (the use of FFTW with a program interface wrapping between two different data structures) was somehow inefficient requiring in the worst case, an allocation of eight times more memory than that effectively required. Of course in most cases the users would not notice or would not even be affected by the existence of this inefficiency. In any case, at this point arises, perhaps, a legitimate question: why the GDT chose an inefficient solution there being other more efficient (i.e. rewrite the GRASS modules with a different data structure) solutions available?

One of the authors of this paper was invited by one of the GRASS developers to discuss this question/problem directly on the GRASS Developers' Mailing List. The following is one of the answers obtained:

> The efficiency issues with the fft() interface are but a single instance of a more widespread issue, namely that the GRASS codebase is extremely large relative to the number of active developers. [GDML, 26 February 2007, http://lists.osgeo.org/pipermail/grass-dev/2007-February/029520.html]

Interestingly, the GDT pointed out that the inefficiency was just a consequence of an organizational problem, rather than the consequence of a purely technical problem: the introduction of the "FFT Interface" was the easiest solution for the problem (less time consuming). Indeed, implementing the "FFT Interface" required limited effort compared with that required for rewriting the data structures. In fact, according to the previous message, the number of active developers in GRASS is limited compared with the dimension of the system codebase.[13] Therefore it might be that in some

[12] A segmentation fault occurs when a program attempts to access a memory location that it is not allowed to access.

[13] 500,000 lines of source code in 2005, and a number of active developers around 10 people.

cases the solutions that require minor effort are preferred, in particular in those situations in which the changes are related to non-fundamental components of the system (as it was with the Fourier Transformation). In addition, in most cases this inefficiency would have not directly affected the GRASS users (only in the worst cases there is indeed a significant inefficiency). Hence, the decision to implement a program interface between the library FFTW and the GRASS commands was the easiest and least time consuming solution. This is, if we follow Law's argument (2004), part of the pre-existing hinterland of GRASS:

> When the switch to FFTW was discussed, efficiency (in terms of memory usage) really wasn't a major factor. The main factors were the amount of developer effort required to make the change and the addition of FFTW as a dependency.
>
> [GDML, 26 February 2007 http://lists.osgeo.org/pipermail/grass-dev/2007-February/029521.html]

The decision to implement the "FFT Interface" was then due to what the GDT considers a major issue (compared to the computer's memory consumption): the limited human resources available for the GDT. In addition the GDT members made us notice that the interface was indeed memory consuming, but that the library FFTW was more efficient than the FFT-NR in term of CPU usage. In 2007 the GDT decided finally to rewrite the data structures of the GRASS programs using the FTTW library (such as i.fft), to make them adhere with the data structure of the library.

6. Discussion

From the case study of this paper, we have seen how the entanglement of a series of social and technical elements "performs" action. The main elements are the following: the software pre-existing the shift to FOSS (NRs) and its data structures and memory allocation processes, the new license, the GRASS developers, the new software (FFTW) and its data structures, the computational power they have (and expect the users to have) and users. All these elements constitute a "socio-materiality" – a seamless web (Bijker, 1995) – that performs a GIS software solution. Empirically, this is how we substantiate the human–artifact relationality in geoIT.

In the case of GRASS, the shift from the non-FOSS Numerical Recipes' implementation of the FFT (contained in GRASS pre-GPL), toward the FOSS library FFTW shows in particular how social choices and pre-existing technical solutions are intertwined in ways that shape the final form of a technology, as the installed base concept itself captures. The choice to implement a program interface as gateway to transform data between two different data structures (from a single array of complex number to two separate arrays, one for the real part and the other for the imaginary part), is indeed a social choice (MacKenzie & Wajcman, 1999), as different solutions were available to programmers. Further, the elimination of the NRs FFT was also a consequence of compliance with legal requirements of software copyright protection and especially requirements of software licenses. This, we argue, shows how actual software development practices contrast with the well-established rhetoric of technical efficiency of algorithms as a principle. It is interesting how seeing efficiency as a possible outcome introduces a shift between the computational and human resources. Socio-technical conditions may lead to inefficient solutions, but with understandable reasons: organizational resources are more scarce then technical, whereas the rhetoric of efficiency originated when computing resources were extremely expensive. Such cultural legacy was reinforced by neighboring disciplines including information systems and management studies. The case of GRASS shows that the contrary might have become true: an idea that was also partly suggested by Raymond (1999) in his well known and influential essay, *The Cathedral and the Bazaar*. Moreover, the influence exercised by pre-existing GRASS software (for instance the legacy of the data structures of programs) illustrates the role of the "installed base": how pre-existing socio-technical elements are an active actor in IT development. This is in line with Chrisman and Harvey's (2004) contribution that different GIS implementations follow a variety of different technological roots. This leads to the need to investigate historically the intertwining of technological and social factors in geoIT.

As any programmer can confirm, this case illustrates a common (but overlooked by research) aspect of GIS development, the different weights that are given to computational and organizational resources, and their entanglement with a preference which is often given to computational resources, especial when – as in FOSS projects – organizational resources are scarce and not materially rewarded. As shown for example by Chrisman (2006) the tradeoff between computer storage and computational power is often an historical problem whose investigation requires historical comparisons and empirical research.

At a first level, we see that the social and the technical perform together: dividing them does not help the analysis, whereas keeping them together enriches our understanding of GIS software development. This consideration comes as a confirmation of the strength of a constructivist approach to GIS. Indeed, in the case described in this paper the efficiency of GIS software proved to be an outcome (effect), not a driver (cause) of socio-technical organization. This supports one of the principles of the social construction of technology proposed by Bijker (1995): that efficiency is indeed an outcome of the seamless web of humans and artifacts. More in the details, the case analyzed in this paper shows the mutual constitution of the social and the technical. Indeed, the interaction between software, algorithms, software licenses, developers and users shapes the actual course of action, driving towards the development of a gateway (the program interface). This is in line with the ANT view, which highlights the relationality of human and non-human actors (Latour, 2005).

7. Conclusions

As final remarks, we highlight how software efficiency has implications on the broader organizational level, so we confirm that we do not claim primacy of labor force scarcity in determining GIS software development. For example long computational time has a number of organizational consequences such as organizational processes' redefinition and IT procurement, among others. Along this line, socio-technical trade-offs can be analyzed and discussed in the light of scalability of geoIT applications. We are not looking into this black-box here.

Rather, we refer back to Orlikowski and Scott (2008) to pinpoint how the approach of socio-material relationality supports an investigation of the rhetorical basis of the efficiency argument. Also, we would like to link this work to Orlikowski and Barley (2001) who claimed that information systems and organizational research can learn from each other. They argue that engineering is oriented to define what works, so functionality is the legitimizing source for this kind of research. Social science approaches are instead closer to the traditional science epistemology, whose aim is explaining and predicting rather than doing. In the latter case, explanatory power – aimed at pushing the boundaries of what is known – legitimizes research. Following this argument we can summarize that mainstream research on geoIT sees technology as a determinant in explaining how things are the way they are, and in prescribing how they can be changed and also their impact in changing society. Constructivism emphasized the role of people and organizations in shaping geoIT. We claim that the human–artifact relationality is a suitable view which avoids both technological

and social determinisms that have characterized the debate (Chrisman, 2005) since very early stages of GIS software development.

Acknowledgements

We thank Markus Neteler and the whole GRASS project. Without them this research would have not been possible. We also thank Vincenzo D'Andrea and Ishwari Sivagnanam for reading early versions of this manuscript. We are indebted with Jan Rigby for the English proofreading.

References

Bijker, W. E., Huges, T. P., & Pinch, T. J. (Eds.). (1987). *The social construction of technological systems: New directions in the sociology and history of technology*. Cambridge, MA: MIT Press.
Bijker, W. (1995). *On bicycles, backelites, and bulbs*. Cambridge: MIT Press.
Bowker, G. C., & Star, S. L. (1999). *Sorting things out: Classification and its consequences*. Cambridge: MIT Press.
Callon, M. (1986). Some elements of a sociology of translation: Domestication of the scallops and the fishermen of St. Brieuc Bay. In J. Law (Ed.), *Power, action and belief: A new sociology of knowledge* (pp. 196–233). London: Routledge and Kegan Paul.
Cheong, P. H., Poon, J. P. H., Huang, S., & Casas, I. (2009). The internet highway and religious communities: Mapping and contesting spaces in religion-online. *The Information Society, 25*(5), 291–302.
Chrisman, N. (2001). *Configuring the users: Social division of labor in GIS software*. <http://chrisman.scg.ulaval.ca/Present/Configuring.pdf>.
Chrisman, N. R. (2005). Full circle: More than just social implications of GIS. *Cartographica, 40*(4), 23–35.
Chrisman, N. R. (2006). *Charting the unknown: How computer mapping at Harvard became GIS*. Redlands, CA: ESRI Press.
Chrisman, N. R., & Harvey, F. (1998). Boundary objects and the social construction of GIS technology. *Environment and Planning A, 30*(9), 1683–1694.
Chrisman, N. R., & Harvey, F. (2004). The imbrications of geography and technology: The social construction of geographic information systems. *Geography and technology* (pp. 65–80). Kluwer Academic Publishers.
Ciborra, C., & Associates (2000). *From control to drift: The dynamics of corporate information infrastructures*. Oxford: Oxford University Press.
Cox, A. (1998). *Cathedrals, bazaars and town councils*. <http://slashdot.org/features/98/10/13/1423253.shtml>.
D'Andrea, V., & De Paoli, S. (2008). Geografia del potere e licenze software: le licenze e la stabilizzazione della conoscenza. In S. Gherardi (Ed.), *Le tecnologie tra lavoro e apprendimento* (pp. 84–112). Bologna: Il Mulino.
Edwards, P. N., Bowker, G. C., Jackson, S. T., & Williams, R. (2009). Introduction: An agenda for infrastructure studies. *Journal of the Association for Information Systems, 10*(5), 364–374.
Ekbia, Hamid R., & Schuurman, N. (2009). Introduction to the special issue on geographies of information society. *The Information Society, 25*(5), 289–290.
Forlano, L. (2009). WiFi geographies: When code meets place. *The Information Society, 25*(5), 344–352.
Free Software Foundation (1991). *GNU General Public License 2.0*. <http://www.gnu.org/copyleft/gpl.html>.
Goodchild, M. F. (1998). Geographic information systems. *Progress in Human Geography, 12*(4), 560–566.
GRASS Developers Mailing List Archive (1991–2010). <http://lists.osgeo.org/pipermail/grass-dev/>.
GRASS Development Team (2010a). Geographic Resources Analysis Support System (GRASS) software, Version 6.4.0. Open Source Geospatial Foundation.
GRASS Development Team (2010b). Geographic Resources Analysis Support System (GRASS) programmer's manual. Open Source Geospatial Foundation.
GRASS Users Mailing List Archive (1991–2010). <http://lists.osgeo.org/pipermail/grass-user/>.
Hanseth, O., & Monteiro, E. (1998). *Understanding information infrastructure*. <http://www.ifi.uio.no/ oleha/Publications/bok.html>.
Hanseth, O., & Monteiro, E. (1997). Inscribing behavior in information infrastructure standards. *Accounting, Management and Information Technology, 7*(4), 183–211.
Harvey, F. (2000). The social construction of geographical information systems. *International Journal of Geographical Information Science, 14*(8), 711–713.
Harvey, F. (2009). Of boundary objects and boundaries: Local stabilization of the Polish cadastral infrastructure. *The Information Society, 25*(5), 315–327.
Homburg, V., & Georgiadou, Y. (2009). A tale of two trajectories: How spatial data infrastructures travel in time and space. *The Information Society, 25*(5), 303–314.
Howell, K. R. (2001). *Principles of Fourier analysis*. Chapman and Hall/CRC.
Kim, S. (2009). Seoul searching: How do mobile communication technologies alter urban mobility? *The Information Society, 25*(5), 353–359.
Latour, B. (1987). *Science in action. How to follow scientists and engineers through society*. Cambridge: Harvard University Press.
Latour, B. (1988). *The pasteurization of France and irreductions*. Cambridge, Mass: Harvard University Press.
Latour, B. (2005). *Reassembling the social: An introduction to actor-network theory*. Oxford: Oxford University Press.
Law, J. (2004). *After method: Mess in social science research*. London: Routledge.
Martin, D. J. (1996). *Geographic information systems: Socioeconomic applications* (2nd ed.). London: Routledge.
MacKenzie, D. A., & Wajcman, J. (Eds.). (1999). *The Social Shaping of Technology* (2nd ed.). Philadelphia: Open University Press.
Monteiro, E., & Hanseth, O. (1995). Social shaping of information infrastructure: On being specific about the technology. In W. Orlikowski, G. Walsham, M. R. Jones, & J. I. DeGross (Eds.), *Information technology and changes in organizational work* (pp. 325–343). Chapman and Hall.
Neteler, M. (2006). Community based software development: The GRASS GIS project. Presentation at the University of Trento, Italy, 30 November, 2006. <http://www.slideshare.net/markusN/community-based-software-development-the-grass-gis-project>.
Neteler, M., & Mitasova, H. (2002). *Open source GIS: A GRASS GIS approach*. Boston: Kluwer Academic Publishers.
Orlikowski, W. (2000). Using technology and constituting structures: A practice lens for studying technology in organizations. *Organization Science, 11*(4), 404–428.
Orlikowski, W., & Scott, S. (2008). The entangling of technology and work in organizations. London School of Economics and Political Sciences, Innovation Group, Working Papers Series 168.
Orlikowski, W. J., & Barley, S. R. (2001). Technology and institutions: What can research on information technology and research on organizations learn from each other? *Management Information Systems Quarterly, 25*(2), 145–166.
Pickles, J. (Ed.). (1995). *The ground truth: The social implications of GIS*. New York: The Guilford Press.
Orlikowski, W., & Iacono, S. (2001). Research commentary: Desperately seeking the "IT" in IT research—A call to theorizing the IT artifact. *Information Systems Research, 12*(2), 121–134.
Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1988). *Numerical recipes in C*. New York: Cambridge University Press.
Raymond, E. (1999). The cathedral and the bazaar. *FirstMonday, 3*(3). <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/578/499> (2 March 1998).
Raymond, E., & Moen, R. (2006). How to ask questions the smart way. <http://floss.syr.edu/Presentations/oscon2006/4_How%20To%20Ask%20Questions%20The%20Smart%20Way.pdf>.
Rivlin, G. (2003). Leader of the free world: How Linus Torvalds became benevolent dictator of Planet Linux, the biggest collaborative project in history. In *Wired magazine*, Issue 11.11, November, 2003. <http://www.wired.com/wired/archive/11.11/linus.html>.
Schowengerdt, R. A. (1997). *Remote sensing models and methods for image processing* (2nd ed). San Diego: Academic Press.
Schuurman, N. (2002). Women and technology in geography: A cyborg manifesto for GIS. *The Canadian Geographer, 46*. <http://www.questia.com/googleScholar.qst;jsessionid= LJQF2rnQ2871h8jrhfhtxfwNv2knK21HyLph2rcG7s0SRG2v1d2L!-2073591579!-797291915?docId=5002516570>.
Schuurman, N., & Pratt, G. (2002). Care of the subject: Feminism and critiques of GIS. *Gender, Place and Culture, 9*(3), 291–299.
Star, S. L., & Bowker, G. C. (2002). How to infrastructure. In L. Lievrouw & S. Livingstone (Eds.), *Handbook of new media* (pp. 230–245). London: Sage.
Star, S. L., & Griesemer, J. (1989). Institutional ecology, 'translations', and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology. *Social Studies of Science, 19*(3), 387–420.
Suchman, L. (2007). *Human–machine reconfigurations: Plans and situated actions*. Cambridge: Cambridge University Press.
US Army CERL (1996). Announcements. <http://web.archive.org/web/19970619195255/www.cecer.army.mil/announcements/grass.html>.
Westervelt, J. (2004). GRASS roots. In *Proceedings of the FOSS/GRASS users conference*, Bangkok, Thailand, 12–14 September, 2004.
Woolgar, S. (1991). Configuring the user: The case of usability trials. In J. Law (Ed.), *A sociology of monsters: Essays on power, technology and domination* (pp. 58–97). London: Routledge.

Glossary

ANT: Actor-Network Theory
FFT: Fast Fourier Transformation
FFTW: Fast Fourier Transformation in the West
FOSS: Free and Open Source Software
GDT: GRASS Development Team
GPL: General Public License
GRASS: Geographic Resources Analysis Support System
GDML: GRASS Developers Mailing List
GUML: GRASS User Mailing List
NRs: Numerical Recipes