Free and Open Source GIS: Will there ever be a Geo-Linux?

Gilberto Câmara, Lúbia Vinhas, Ricardo Cartaxo Modesto de Souza

National Institute for Space Research, INPE, Av dos Astronautas 1758, São José dos Campos, SP, Brazil

Abstract. This paper examines the constraints that limit the large-scale adoption of open source GIS. Although the open source GIS community has already achieved relevant results, their products have a small market share. There is equivalent to Linux and Apache in the open source GIS scene. We try to explain why this happens, by considering some factors that control the evolution and adoption of open source software. Our view is that the community effort is split in many different systems, not allowing a dominant solution to come forth. Thus, none of the current open source GIS has the potential to be a disruptive technology. Then, we consider a future scenario where most public geospatial data will be available as open access policy. This scenario is becoming more probable given recent data policy decision in Europe, USA and other countries. In this scenario, there is a major chance for a disruptive open source GIS to appear.

1. Introduction

Free and open source software (FOSS) has moved from being a specialist niche to the mainstream. The FOSS community has grown considerably, as illustrated by its multiple workshops and meetings. In some areas of information technology (supercomputers, data servers, and scripting languages) the leading market solutions are open source. Supercomputers and data servers use Linux as their operating system and companies such as IBM have hundreds of programmers in their Linux teams. All major scripting languages (Perl, Phyton, Lua, Ruby) are FOSS. However, there are some areas where FOSS has failed to gain a substantial market share. Document production software is dominated by a single proprietary solution, despite the community efforts for building solutions such as OpenOffice. Nagy et al [1] examine the rate of adoption of FOSS in the business community. Their estimates indicate that the Apache web server has about 50% market share and Linux is used as the operating system of about 30% of servers. By contrast, the Open Office suite has a marginal market share of about 3% of the desktop. As the authors state: *"This pattern suggests that there may be significant barriers to open source software adoption among some sectors of the user populations"*. Clearly, some applications areas are more prone for FOSS adoption than others.

In the area of GIS (geographical information systems), the commercial market is dominated by one company (ESRI), which has more than 30% of market share considering software and services. By contrast, the free and open source GIS scene (FOSS4G) is fragmented in diverse niches. Recent surveys of open source GIS provide a good overview of the different products [2-5]. The view expressed in these papers is generally positive of the accomplishments of OSS GIS community. Ramsey [4] remarks that *"existing products are now entering a phase of rapid refinement and enhancement"*. Steiniger and Bocher [2] point out that *"[FOSS4G] projects have reached a mature stage and the software offers a multitude of functionalities."* Nevertheless, none of the many available open source desktop GIS has a significant market share. In this chapter, we examine the reasons that limit large-scale adoption of FOSS4G. Thus, the question we address in this paper is: *will there ever be an FOSS4G equivalent to Linux or Apache? If so, how will a Geo-Linux appear?*

Thus, this paper is concerned with examining the structural constraints that limit the adoption of FOSS4G by a large number of users. We start by considering some factors that control the evolution and adoption of FOSS. By examining the reasons why Apache has 50% market share and OpenOffice only 3% [1], we can learn some lessons which are useful for the future of FOSS4G. Then, we try to explain why the FOSS4G scene is split up in many different systems. Finally, we propose some possible futures for FOSS4G development.

2. Factors for User Adoption of FOSS

This section proposes some structural reasons that influence users to select a FOSS product. There are many papers that analyze the nature and evolution of FOSS [1, 6-13]. Most of these papers discuss the organization and motivation of FOSS teams and the economics of open source software development. We believe that the structural reasons that govern the users' adoption of a FOSS have not been discussed in detail in the literature [14]. In the following discussion, we focus on the factors that users need to consider for adoption of FOSS. On purpose, we left out an obvious factor: lower cost. Although promoted by FOSS advocates as a major drive for adoption of open source technology, lower cost by itself is not sufficient for users in most cases. This is verifiable empirically by considering the low adoption rate of OpenOffice. Thus, we propose the following factors as contributing to the adoption of FOSS.

Degree of modularity. One of the major factors mentioned in favour of open source adoption is capacity of users to modify a FOSS to suit their needs [15]. In practice, such adaptability is not easy to achieve since it requires a modular software organization. The more a FOSS is modular, the easier it is to adapt it to the needs of different organizations. Each software product has a *periphery to kernel ratio* that constrains the potential for modularization, since the kernel needs a tightly organized and skilled programming team. An operating system such as Linux has a well-defined kernel for process control and a periphery consisting of programs such as device drivers, applications, compilers and network tools. Differently, database management systems have a kernel of integrated functions (parser, querier, scheduler, and optimizer) and a much smaller periphery. A high *periphery to kernel ratio* means a more modular software. The more the kernel is dominant in a software product, the less modular it is, and the less potential contributors it has. One good example is the R suite of statistical tools [16], whose kernel consists of a small and well-designed library to deal with scientific data. Developing a new application using R is very simple. Many different researchers have developed a large set of statistical software that makes R very popular worldwide.

Shared conceptualization. A good conceptual design is a crucial part of any successful software project. Faulty designs are a major cause of failures in software projects [17]. The design problem is even more important for FOSS. If programmers scattered in different places are to communicate well, they need to share the same conceptual view. This conceptual view is difficult to capture in written documents, and is much easier to achieve when there is a prior common background. This explains why many successful projects rely on existing designs. We call this '*shared conceptualization*' [14]. Again, the R suite of statistical tools is a good example. R has a simple and well-understood conceptual design, based on the S-PLUS proprietary product [18]. Their innovative contribution lies not in their design, but on the analysis functions that scientists develop using these environments. The two main conditions for shared conceptualization to happen are:

1. The *post-mature* perspective [19]: a private company develops a software product, for which it holds the intellectual property rights. As the product becomes popular, its functionality and conceptual model becomes well settled, and it becomes part of the "public commons". The popularity and usability of the software motivates other institutions to develop a public domain equivalent, as in the case of R.
2. The *standards-led* perspective [19]: standards consolidate a technology and allow compatible solutions from different producers to compete in the marketplace. An example is the SQL database standard, which has motivated products such as mySQL and PostgreSQL. Another example is the POSIX standard for operating systems, which has served as guidance to Linux.

Design and maintenance challenge. Many users and companies use FOSS because it would be hard for them to design and maintain a proprietary solution. For example, consider a hardware company that needs an operating system and has to choose between building its own or adopting an OSS. Knowing that the Linux kernel has over 12 million lines of code and works well, the company´s managers have a large incentive of using Linux. Indeed, the high cost of maintaining a proprietary Unix-like operating system is the main reason why IBM, SGI, and HP switched to Linux. Munga et al [9] point out that IBM has made FOSS a core part of their business strategy, and there are an estimated 600 programmers working full-time as part of their Linux team. The same factor explains the adoption of the Apache web server. Designing and maintaining a reliable and fast web server is difficult. The Apache HTTP Server today powers nearly 112 million websites world-wide. Apache´s proven reliability and large user base deters commercial companies or other OSS developers from developing alternatives. Thus, when an OSS provides a service that would be hard or costly to replicate, users have a large motivation for using it, instead of developing their own solutions or using proprietary software.

Code stability and avoidance of forking. Programmers use the word "forking" to describe the situation when different groups develop incompatible products, starting from the same code. The avoidance of forking and the resulting code stability is an important factor in FOSS adoption by users. To show the negative impact of forking, consider the parallel evolution of Linux and FreeBSD. From 1977 to 1995, the Computer Systems Research Group (CSRG) of the University of California at Berkeley developed a Unix-like operating system, called the Berkeley Software Distribution (BSD). After the University stopped this activity in 1995, the code was released as open source and taken up by a community of programmers. Differences of opinion among the code developers led to fragmentation of the BSD-Unix community into FreeBSD, OpenBSD, and NetBSD camps [1]. By some accounts, the BSD-derived operating systems were more advanced than Linux in the early 2000s [20]. However, the disputes between the three BSD communities were damaging to their reputation. By contrast, Linus Torvalds has always maintained a tight control over the Linux kernel, an attitude that ensured code stability and user trust. Forking is thus one of the reasons why BSD failed to reach a wide user base, while Linux emerged as the leader of the open source operating systems [1].

Path Dependence and Lock-in Effect. Path dependence is an established property of technology. Although definitions vary, the idea of path dependence is that the current status of a technology (e.g., the dominant software at the desktop) has evolved as a consequence of the initial configuration (e.g., IBM´s selection of Microsoft for developing the operating system of its PC) [21]. According to Page [22] path dependence has different causes including *increasing returns*, *self-reinforcement*, *positive feedbacks* and *lock-in*. Increasing returns means that the more a choice of a product is made, the greater its benefits both to the producer and consumer. Self-reinforcement means that when a costumer makes a choice, he gets connected to other costumers that encourage that choice to be sustained. With positive feedbacks, the benefit of making a choice is increased when others do the same. The "lock-in" effect occurs when a customer is dependent on a single supplier for products and services and cannot move to another vendor without large switching costs. The lock-in effect is well-known in the software industry since the customer may become dependent on proprietary data formats or interfaces, and high switching costs might prevent the change to another product [23].

Avoiding the "lock-in" effect has been proposed by some authors as an important motivation for using open source. According to this view, users of FOSS would have the capacity to switch between different vendors and thus reduce their dependence on a single manufacturer. Thus, a Linux user can choose between hardware from different manufacturers, where an IBM mainframe user is constrained in his future choices. However, like all technology products, FOSS is also affected by path dependence, mostly by *increasing returns*. When a user chooses Linux over

FreeBSD, he reinforces the notion that Linux is the best choice for a FOSS operating system. Considering that many FOSS products rely on community support for helping novice users and producing documentation, path dependence is very relevant to FOSS. In the long run, increasing returns encourage a single FOSS solution to be dominant in each market segment. By contrast, the existence of many Linux distributions harms its expansion at the desktop. Non-technical users are discouraged by the alternatives and fear for their long-term endurance. We consider that the convergence to a single distribution is a necessary condition for Linux to reach a significant market share of the desktop users.

Disruptive potential. An important factor in the adoption of a FOSS is its disruptive potential, considered as its capacity to provide a novel functionality that can have long-term consequences. Such is the case of Linux. Before Linux emerged as a viable alternative, most Intel-based computers ran Windows, and each major UNIX workstation builder (such as IBM, DEC, SUN, HP and SGI) had its own line of CPUs. However, these companies had a hard time competing with Intel, as the scale of investments each successive line of CPUs increased. Maintaining Moore´s Law is expensive. So, the availability of a UNIX-like operating system running on Intel chips caused the demise of non-Intel alternatives such as IRIX and Tru64 Unix, the transformation of Solaris into a FOSS, and the reduction of the market for HP-UX and AIX. Even IBM, which still maintains its AIX operating system, is making substantive investments on Linux [24]. Thus, Linux was a disruptive technology that ultimately changed the whole UNIX-like scene.

Documentation. Another significant factor that limits FOSS large-scale adoption is user documentation. Users have different learning processes, and thus it is necessary to produce different types of manual and tutorials to help them. In most cases, producing such documentation is beyond the capacities of FOSS teams. That is one of the reasons why FOSS is more popular with computer-savvy experts that with the non-technical users.

Community support. Having a strong support from an established community helps to decrease the path dependence effect of proprietary software on users. As an example, the PROJ4 library has an active community that helps to support its use by different software products.

3. The Current FOSS4G Scene

We now consider how the factors for adoption for FOSS relate to the current FOSS4G scene, taking as our reference the works of Steiniger and Bocher [2] and Ramsey [4]. Based on these reviews, we can divide the FOSS4G scene into five clusters:

(a) *Java-based visualization and analysis software*: programs written mostly in Java that enable the visualization of spatial data and provide functions for analysis of such data. They include uDig, gvSIG, KOSMO, and OpenJUMP and its derivatives. These software also provide functions for handling vector data.

(b) *C and C++ -based visualization and analysis software*: programs for visualization and analysis of spatial data using the C and C++ programming languages. These include QGIS, TerraView, and MapWindow. As in case (a) above, these software are capable of handling vector data.

(c) *Image and Raster data analysis software*: programs for analysis of (mostly) raster data, such as images and digital terrain models. These include OSSIM, GRASS, SAGA, SPRING, and ILWIS. GRASS and SPRING also provides significant capabilities for working with vector data.

(d) *Database interfaces*: APIs that provide support for access to object-relational database systems to handle vector data. The best example is PostGIS [25]. Other example is the TerraLib library, designed for handling large-scale environmental data. It can access PostGIS and other DBMS's and also allows access and storage to raster data [26]. Other notable effors include GDMS[1] and Hybernate spatial[2].

(e) *Supporting libraries*: these libraries support functions which are needed by any GIS, and are used by many of the current front-end implementations. They include GDAL/OGR, PROJ4, GEOS, JTS (Java Topology Suite) and Geotools.

There are many additional differences between the above-mentioned software. Some support the OGC standards, others are file-based, and the analysis functionality varies greatly. In terms of visualization, most pro-

[1] http://hal.archives-ouvertes.fr/hal-00358740_v1/
[2] http://www.hibernatespatial.org/

vide similar capabilities. For vector data analysis, no FOSS4G package provides an equivalent quantity of functions as those available in the leading proprietary GIS (ESRI´s ArcGIS). For raster data analysis, the functionalities of FOSS4G products approach those of proprietary software, although in some niche areas (digital photogrammetry, object-based image analysis) proprietary software still has the lead.

The lack of accurate surveys makes it difficult to assess the impact of FOSS4G on the market. We have to rely on indirect assessments, such as the difference in user attendance in FOSS4G and ESRI´s conferences, and the small number of large-scale applications that use FOSS4G. Such scant evidence suggests that none of the available FOSS4G products has reached a significant proportion of desktop users, which continue to rely on proprietary software. So far, no desktop GIS has shown potential for breaking the 'lock-in' effect which proprietary software products have on users. This is a direct consequence of the fragmentation of the community. To understand more, we need to consider both the achievements and challenges to FOSS4G, as follows.

3.1 Achievements of the FOSS4G community

The most important achievement of the community is intangible: FOSS4G exists and is being used worldwide for real-world applications. Many case studies have been presented at the FOSS4G Conference and similar venues. These cases show that the skills required to build a FOSS4G application are similar to those required to use proprietary software[3]. In general, the TCO (total cost of ownership) of a FOSS4G application is significantly less than that of a proprietary application. Personnel salaries and hardware costs are similar, and software costs are much reduced.

A large part of the credit for the success of FOSS4G product is due to the PostGIS project and the activities of the GRASS user community. The GRASS user community includes users from all parts of the world, that actively participate in mailing lists and user conferences[4]. As for PostGIS, we concur with Ramsey [4]: *"The strength of PostGIS is that it has become the standard spatial database backend for all the other open source*

[3] For some metrics on the subject, please see the discussion http://lists.osgeo.org/pipermail/conference_dev/2010-March/001029.html
[4] See for example http://gisws.media.osaka-cu.ac.jp/grass04/

GIS tools." Indeed, programs such as uDig, gvSIG, KOSMO, OpenJUMP, and TerraLib rely on PostGIS as their database support.

PostGIS and the underlying PostgreSQL DBMS have proven capabilities of support industry-grade applications. Take for example TerraAmazon, Brazil's national database for monitoring deforestation in Amazonia. TerraAmazon is based on TerraLib that in turn uses PostGIS with added support for raster storage in PostgreSQL [27]. The database currently stores more than 3 million complex polygons and 600 GB of full resolution satellite images, using the TerraLib pyramidal resolution schema.

Another area of achievement is that of application libraries. Products such as GDAL/OGR, PROJ4 and GEOS are being used by most of the FOSGIS community, due to their stability and quality. These libraries are well-designed and solve important issues in GIS design, namely: interface to data formats (GDAL), cartographic projections (PROJ4) and topological predicates and operations (GEOS and JTS).

3.2 Challenges to the FOSGIS Community

The authors consider the division between the 'C/C++', 'Java', and 'raster' clusters as the major impediment for the growth of FOSS4G solutions in the market place. This fragmentation is a good example of the dreaded forking pattern which keeps users away from FOSS. In an ideal but unlikely situation, if all FOSS4G developers would agree to use a single environment, the community could build a system that would surpass the current proprietary software in quality and functionality. As the situation stands, there are too many systems offering similar functions, whereas none has the power to become dominant.

Modularity. Most desktop GIS are monolithic applications, where the visualization actions are closely tied to the handling of interface events. The design choices for GUI design and event handling are hard to share across a community of programmers. There are many incompatible options for GUI design (Qt, GTK+, SWT, SWING) and there are plausible reasons for choosing any of them. Once a GUI base library is chosen, the desktop GIS code becomes hard-wired on that library and is difficult for outsiders to make effective contributions to the software. Because of the hard-wired links between their interface and the underlying GUI library, desktop GIS applications tend to be monolithic and non-modular.

Leadership. Good programmers are notoriously strong-minded and respect only those they view as superior to them. As Fitzgerald [11] observes: "*to lead an OSS project really successfully, the leader needs to be a 'code god' who inspires others and whose ability and authority is beyond discussion.*" Most of the current FOSS4G is being developed independently by like-minded teams, who had limited contact with their peers. While there are recognized programmers in the community, there is no established leadership in the sense of Linus Torvalds and Richard Stallman. The project leaders of the different desktop GIS products have thus little incentive for merging their activities with those of similar initiatives. The main gathering of the FOSS4G community (the annual FOSS4G Conference) was established after most of the desktop GIS projects were already underway. The FOSS4G Conference serves as an important contact point for members of the community, but so far it has not promoted any convergence of FOSGIS technology. The authors hope that the OSGEO organization (founded in 2006) can act as a catalyst. In an ideal world, OSGEO could support a large scale communitarian effort that would produce a FOSS4G product that could outperform today´s proprietary solutions.

Standards and shared conceptualization. Most desktop GIS look similar, since they share a common paradigm for their interface (that of the proprietary software ArcView). Most of them also share the OGC standard interfaces for web services and vector data. However, the OGC effort on GIS standardization is commendable but incomplete. There has been significant progress in OGC for vector data storage, database query, and web services. However, there are no mature standards for data analysis, image processing, map algebra, digital terrain modelling, and spatial statistics[5]. The lack of shared conceptualization for analysis and processing functions has led to incompatible products. Each GIS development team has developed his own implementation of these functions, leading to further incompatibility between the products. Applications such as GRASS, SAGA, SPRING, and ILWIS use many different concepts. Indeed, proprietary image processing software such as ENVI, ERDAS and PCI also have significant differences. As a result, each software has a slow learning curve which creates a lock-in effect on its users. By contrast, the spatial statistics community that uses the R software [16] has decided to join forces and develop a single product (called *R-spatial*) that provides most of the functionality they need [28]. A further gray area concerns storage of raster data

[5] Such lack of standards has motivated many developers to come up with new products, such as OGRS team at IRSTV Nantes (France) and the TerraLib team at INPE (Brazil).

in DBMS. Although ORACLE already provides support for storing raster data in its proprietary products, there is much debate on the FOSS4G community about the advantages and problems of storing raster data in a free spatial DBMS such as PostGIS. This is another area which would benefit from a standard that could be followed by the community.

Code stability and avoidance of forking. The large number of FOSS4G products indicates that designing a desktop GIS is relatively easy, within the reach of a small team of programmers. This is bad news. It is too easy to start a new project, and this reduces the motivation for adoption a solution developed elsewhere. Some FOSS4G projects, such as JUMP and GeoTools, have gone through phases of forking and merging with other projects. This is negative for code stability and tends to restrain user adoption. Part of the problem has to do with language choice. Currently, most GIS programmers have adopted Java as their preferred development environment. However, many existing products (such as GRASS, SPRING, TerraView, ILWIS) have lots of code already developed and tested using C or C++. This causes an impasse when trying to expand a Java-based visualisation and vector handling software into a more complete GIS. The solution here is to develop stable software libraries that could provide all the functionalities of products such as GRASS or SPRING and that could be used by different products. The PostGIS, GDAL, PROJ4 and GEOS libraries are good examples of code stability and continuous improvement that need to be complemented and extended.

Innovation. To stay ahead of the market, companies have to innovate continuously. As Andrew Grove of Intel said, "*only the paranoid survive*". The GIS market is currently full of activities related to mobile devices and to web services. However, FOSS4G products tend to be conservative and focused on the desktop, based on the map metaphor. By contrast, the commercial vendors (especially ESRI and Google) are investing heavily in mobile GIS applications. Considering the money commercial companies spend on innovation, FOSS4G developers would need to be united behind a small number of innovative products to be able to compete.

Disruptive potential. PostGIS is the FOSS4G product with greatest potential to be a disruptive technology and to change the GIS market. Should PostGIS evolve to support different types of geospatial data needed by emerging GIS applications, such as images, DTMSs, geo-sensor data, and location-based services, it could provide the support FOS GIS needs to provide strong competition to proprietary software. We will return to this issue in the next section. In short, while PostGIS has a good potential to be an important player in the user GIS market, the other desktop GIS lack the momentum to become dominant, and the raster data processing solutions are incompatible and hard to combine. While this split persists, the leading commercial companies continue to innovate and to keep or increase their market share.

4 GIS in the 21st Century: What can become of FOSS4G?

The preceding sections tried to show why current there is currently no good candidate for a Geo-Linux. While PostGIS has a good momentum to provide the services needed by the different types of geospatial applications, there is no dominant solution in the desktop, nor there are emerging applications for mobile devices and web services. Does that mean that there will never be a Geo-Linux? As the baseball manager Yogi Berra said, "*The future ain't what it used to be*". If the FOSS4G community is to design products that reach a significant market share, it needs to learn the lessons from Linux and Apache. We need products with disruptive potential, not limited to reproducing designs from proprietary products, providing truly innovative capabilities. In this perspective, this section examines two areas which have the potential to bring forth disruptive applications: *Web services* and *data-intensive GIS*.

Consider Web services. In the early 2000s, the potential of the Web-enabled spatial applications became clear to the GIS community. Their reaction was build software for data visualization and browsing on the Web (such as MapServer) and to establish a set of standards, including WCS, WMS and WFS from the Open Geospatial Consortium (OGC). The abstractions encapsulated in OGC´s Web standards deal mostly with a non-cooperative environment. This approach underestimates the Web´s potential for innovation and considers it just a medium for visualization of remote information. Using OGC´s standards, users have access to information produced by others, mostly for visualisation. The user is thus a passive consumer of information produced elsewhere [29]. Using social networks in the Internet, the FOSS4G community could build collaborative systems that go beyond the simple OGC abstractions and support cooperation and interaction. Although there are some exceptions, such as Open Street Map, most FOSS4G applications are conservative in their use of the Web´s potential.

The Spatial Web GIS is more and more being dominated by Google Maps and Google Earth. Google developers come from a different mindset than the map-centred GIS community. They recognized the value of supplying the base location layer (vectors and images) as a seamless data set, and of providing APIs for extensibility. The Google Maps API allows programmers to embed Google Maps in their web pages and to add content to the base layers. The KML format and the Google Earth API allow the user to add maps, images, routes, and other spatial documents to Google Earth, thus easing the task of creating a Web GIS server. Such flexibility and performance comes at a price: users must comply with the IP restrictions set by Google. From a FOSS4G viewpoint, Google´s potential for dominance of the Spatial Web is dismaying. Should it happen, the control of the Spatial Web by a proprietary solution will limit both the development of FOSS4G and reduce the incentive for sharing data, either voluntary or government produced. It is time the FOSS4G community learns the "Google lesson": *Those that control the data control the future of the Spatial Web*. Google Maps and Google Earth show what can be done when only one company has access to key spatial data, such as street networks and remote sensing images.

The limitations of RDBMS and SQL for handling geospatial data have been recognized long ago [30]. Nevertheless, object-relational databases have been adopted by the GIS community as the primary way to organize collections of spatial data. Using PostGIS or its proprietary alternatives, GIS users have built huge datasets, each modelled using a different conceptual schema. When these datasets were created in the early to mid 2000s, the prevailing business model was to have public data sold on a cost-recovery basis. This business model is changing and national mapping agencies are moving towards an open access model. In April 2010, the UK Ordnance Survey launched a service offering free and unrestricted access to most of its map data. Similarly, the US Geological Survey has released its complete LANDSAT imagery archive. Many other mapping agencies and image providers have released their datasets or are planning to do so. When achieved, full open access to spatial data will bring a complete change on the way GIS technology is used and will provide a unique opportunity for FOSS4G: *the emergence of data-intensive GIS*.

We define *data-intensive GIS* as a software environment capable of handling distributed geospatial data sets, each with its own archival policy. These datasets include spatial databases in OGC format, WCS, WFS and WMS servers, and semi-structured data in different formats (flat files, indexed collections, relational tables). A data-intensive GIS application would be able to use these data sets in a coherent way combining different sources and making appropriate semantic inferences whenever needed. A data-intensive GIS would use techniques that go beyond what the current extensions of SQL allow. For example, such techniques would allow multiple passes over the same data to compute spatial algorithms, a task with is cumbersome to perform using SQL queries. They would also combine different types of spatio-temporal data in a way not possible for SQL and its extensions.

Consider a situation where all government-produced census data, streets and routes, and satellite imagery would be fully accessible on the Web, based on copyleft licences such as GPL or Creative Commons. The FOSGIS community could design, build and maintain applications that would handle distributed spatial data. They would use powerful inference techniques to make sense of the data they collect. These data-intensive GIS application could disrupt the current market and change the way we deal with geospatial information.

If the above proposal seems far-fetched, consider that similar techniques are already available. Again, Google is the leading innovator. Google´s Map-Reduce technique has shown to be efficient for processing multi-terabyte datasets in parallel [31]. A Map/Reduce job has two steps. First, a master node splits the input dataset into independent chunks and passes each of them to a secondary node for processing. When the master node receives the results, sorts them, and passes which are processed by worker nodes in a completely parallel manner. The master node then takes the answers to all the sub-problems and combines them to answer the problem it was originally trying to solve. Typically both the input and the output of the job are stored in a file-system. More than 10,000 distinct programs have been implemented using Map-Reduce at Google, including algorithms for large-scale graph processing, text processing, machine learning, and statistical machine translation [31]. Even the defendants of RDBMS recognize that there are processing tasks that are well-suited for Map-Reduce [32]. The Map-Reduce paradigm is suited for investigation as one of the alternatives to SQL queries for handling geospatial data.

In resume, the emergence of open access policies for geospatial data will provide an opportunity for the FOSS4G community to develop innovative and disruptive applications. To succeed, GIS developers will have to overcome a number of challenges. These include modelling the semantics of spatial concepts, understanding change in space and time, and developing information extraction methods for massive data sources [29].

The success of information technologies such as Google´s shows that these challenges are hard, but not impossible to take on. Should the FOSS4G community succeed in producing innovative solutions for data-intensive GIS, it is likely that a Geo-Linux will finally emerge.

5. Concluding Remarks

This paper discusses the current situation of FOSS4G. We recognize that the FOSS4G community has already achieved relevant results. There are many good desktop FOSS4G available, which support the basic tasks of retrieval and visualization. The various OS raster data processing software have lots of functions. However, FOSS4G products still have a small market share. In this paper, we try to explain why this happens. Based on the properties of open source software in general, we argue that there is excessive fragmentation on the FOSS4G scene. Also, current FOSS4G are not modular enough to support large development teams. Furthermore, there is a lack of common standards for many important GIS applications, such as map algebra, spatial statistics and image processing, leading to similar but incompatible products. More important, no current FOSS4G has a disruptive potential that could unbalance the market. We also indicate that there are structural reasons in GIS design that contribute to produce this status quo.

Then, the authors examine two areas that are relevant for the future of GIS: *Web services* and *data-intensive GIS*. We argue that the trend towards open access geospatial data brings an opportunity for FOSS4G. We could develop disruptive GIS that could support data-intensive applications, combing data from diverse sources and allowing the user to overcome semantic barriers. If the FOSS4G community succeeds in developing such an innovative product, we could finally have a Geo-Linux. The enormous potential of a dedicated community would be at last realized.

Acknowledgments

The first author thanks Erwan Bocher and his team at Nantes for organizing the OGRS 2009 conference and inviting him to deliver a keynote speech, which led to this paper. The authors also thank the Terralib development team at INPE, especially Antonio Miguel Monteiro, Gilberto

Ribeiro, and Karine Ferreira for many hours of inspired discussion. Gilberto Camara's work is partially funded by CNPq (grant PQ 550250/2005-0) and FAPESP (grant 04/11012-0).

Bibliography

1. Nagy, D., Yassin, A.M., Bhattacherjee, A.: Organizational adoption of open source software: barriers and remedies. Commun. ACM **53** (2010) 148-151
2. Steiniger, S., Bocher, E.: An Overview on Current Free and Open Source Desktop GIS Developments. International Journal of Geographic Information Science **23** (2009) 1345-1370
3. Steiniger, S., Hay, G.J.: Free and Open Source Geographic Information Tools for Landscape Ecology. *Ecological Informatics* (2009)
4. Ramsey, P.: The State of Open Source GIS. Refractions Research, Victoria, BC, CA (2007)
5. Hall, G., Leahy, M. (eds.): Open source approaches in spatial data handling. Springer, Berlin (2008)
6. O´Reilly, T.: Lesson from Open Source Software Development. Commun. ACM **42** (1999)
7. Benkler, I.: Coase's Penguin, or, Linux and The Nature of the Firm. Yale Law Journal **112** (2003)
8. Hertel, G., Niedner, S., Hermann, S.: Motivation of software developers in the F/OSS projects: an Internet-based survey of contributors to the Linux kernel. Research Policy **327** (2003) 1159-1177
9. Munga, N., Fogwill, T., Williams, Q.: The adoption of open source software in business models: a Red Hat and IBM case study. Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists. ACM, Vanderbijlpark, Emfuleni, South Africa (2009)
10. Shibuya, B., Tamai, T.: Understanding the process of participating in open source communities. Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development. IEEE Computer Society (2009)
11. Fitzgerald, B.: A Critical Look at Open Source. IEEE Computer **37** (2004) 92-94
12. Mockus, A., Fielding, R., Herbsleb, J.: Two case studies of open source software development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology **11** (2002)
13. Ghosh, R.A., Rudiger Glott, Kreiger, B., Gregario Robles: The Free/Libre and Open Source Software Survey and Study—FLOSS Final Report. International Institute of Infonomics, University of Maastricht, Maastricht, The Netherlands (2002)
14. Câmara, G., Fonseca, F.: Information Policies and Open Source Software in Developing Countries. Journal of American Society of Information Science and Technology **58** (2007) 121-132
15. Weber, S.: The Success of Open Source. Harvard University Press, Cambridge, MA (2004)
16. Ihaka, R., Gentleman, R.: R: A Language for Data Analysis and Graphics. Journal of Computational and Graphical Statistics **5** (1996) 299-314
17. Brooks, F.: No Silver Bullet: Essence and Accidents of Software Engineering. IEEE Computer **20** (1987) 10-19
18. Chambers, J.M.: Programming with Data. Springer-Verlag, New York, NY (1998)
19. Câmara, G., Onsrud, H.: Open Source GIS Software: Myths and Realities. In: Esanu, J.M., Uhlir, P.F. (eds.): Open Access and the Public Domain in Digital Data and Information for Science: Proceedings of an International Symposium. The National Academies Press, Washington (2004) 127-133

20. Yu, L., Schach, S.R., Chen, K., Heller, G.Z., Offutt, J.: Maintainability of the kernels of open-source operating systems: A comparison of Linux with FreeBSD, NetBSD, and OpenBSD. Journal of Systems and Software **79** (2006) 807-815
21. David, P.: Path dependence: a foundational concept for historical social science. Cliometrica **1** (2007) 91-114
22. Page, S.: Path dependence. Quarterly Journal of Political Science **1** (2006) 87-115
23. Ruttan, V.: Technology, Growth and Development. Oxford University Press, New York (2001)
24. Samuelson, P.: IBM's pragmatic embrace of open source. Commun. ACM **49** (2006) 21-25
25. Santilli, S., Leslie, M., Hodgson, C., Ramsey, P., Lounsbury, J., Blasby, D.: PostGIS Manual - Version 1.3.2. Refractions Research, Victoria, CA (2007)
26. Câmara, G., Vinhas, L., Ferreira, K., Queiroz, G., Souza, R.C.M., Monteiro, A.M., Carvalho, M.T., Casanova, M.A., Freitas, U.M.: TerraLib: An open-source GIS library for large-scale environmental and socio-economic applications. In: Hall, B., Leahy, M. (eds.): Open Source Approaches to Spatial Data Handling. Springer, Berlin (2008) 247-270
27. Freitas, U., Ribeiro, V., Queiroz, G.R., Petinatti, M., Abreu, E.: The Amazon Deforestation Monitoring System: A Large Environmental Database Developed on TerraLib and PostgreSQL. OSGEO Journal **3** (2007) 70-75
28. Bivand, R., Pebesma, E., Gómez-Rubio, V.: Applied spatial data analysis with R. Springer Verlag (2008)
29. Câmara, G., Vinhas, L., Davis, C., Fonseca, F., Carneiro, T.: Geographical Information Engineering in the 21 st Century. In: Navratil, G. (ed.): Research Trends in Geographic Information Science. Springer, Berlin (2009) 203-218
30. Egenhofer, M.: Why not SQL! International Journal of Geographical Information Systems **6** (1992) 71-85
31. Dean, J., Ghemawat, S.: MapReduce: a flexible data processing tool. Commun. ACM **53** (2010) 72-77
32. Stonebraker, M., Abadi, D., DeWitt, D.J., Madden, S., Paulson, E., Pavlo, A., Rasin, A.: MapReduce and parallel DBMSs: friends or foes? Commun. ACM **53** (2010) 64-71