Journal of the American Society for Information Science and Technology (JASIST), in press, 2006

Information Policies and Open Source Software in Developing Countries

Gilberto Camara
Image Processing Division (DPI)
National Institute for Space Research (INPE)
Av dos Astronautas, 1758
São José dos Campos SP 12227-001
Brazil
Phone: +55-12-3945-6499
Fax: +55-12-3945-6460
E-mail: gilberto@dpi.inpe.br

Frederico Fonseca
School of Information Sciences and Technology
The Pennsylvania State University
307E Info Sciences and Technology Building
University Park, PA 16802-6823
U.S.A.
Phone: (814) 865-6460
Fax: (814) 865-6426
E-mail: fredfonseca@ist.psu.edu

Information Policies and Open Source Software in Developing Countries

Abstract

Many authors propose that open source software (OSS) is a good strategy to bring information and communication technologies to developing countries. Nevertheless, the use of OSS needs to be more than just adopting Linux as the standard for operating systems. Adoption of OSS is not only a choice of software, but also a means of acquiring knowledge. Developing countries have to use OSS as a way to gain knowledge about the technology itself and as a way of creating technology products that fit their specific needs. In this paper we introduce a model of OSS based on its essential characteristics to understand how developing countries may use OSS to achieve their development goals. We argue there are two defining properties of any open source software. The first property is the potential for shared conceptualization and the second is the potential for modularity. By assessing how each OSS project satisfies these two conditions, we build a taxonomy for open source projects. This taxonomy will help the development of more sensible policies to promote the use of open source in developing countries.

Keywords: Open Source Software, Developing countries, Information and communication technologies, Software development, Information policies

1. INTRODUCTION

Information and communication technologies (ICTs) are at the center of recent transformations in our society. Although the changes take place mostly in industrialized nations, some developing countries are also becoming aware of the potential for change that ICTs bring with themselves. There is an increasing consensus the arrival of ICTs may be a good opportunity for developing countries to reach their development goals. Nevertheless, there are some cases in developing countries in which ICTs are not being effective in improving the lives of people (Mansell & Montalvo, 1998). In developing

countries, even in areas of the economy in which technology could play a fundamental role, the rate of failure of ICTs can reach almost 50% (Ehikhamenor, 2003).

There are many suggestions of feasible strategies for implementing ICTs in developing countries. A topic of particular interest is adopting open source software (OSS) as a means of reducing licensing costs and of promoting indigenous technological development by having access to the source code of these products. A recent document on intellectual property rights and international development commissioned by the government of the United Kingdom underpins such policies. The main recommendation is that "*developing countries and their donor partners should review policies for procurement of computer software, to ensure that options for using low-cost and/or open-source software products are properly considered and their costs and benefits carefully evaluated*" (Barton et al., 2002). Other reasons for adoption of OSS in developing countries include avoiding being hostage to proprietary software (UN, 2004), advancing knowledge more quickly (UN, 2003), and helping to set up an information economy (Weerawarana & Weeratunga, 2004).

Open source software is thus considered to have a potential impact for knowledge acquisition by developing nations. As Weber (2004) points out, combining free software tools with the technical workforce available in developing countries can enable technology transfer. He states that *"the essence of open source is not the software. It is the process by which software is created"* (Weber, 2004 p.56). He expects OSS to have far-reaching effects: *"Of course information technology and open source in particular is not a silver bullet for long–standing development issues; nothing is. But the transformative potential of computing does create new opportunities to make progress on development problems that have been intransigent"* (Weber, 2004 p. 254).

As seen above, many authors express hopes about the potential and expected impact of OSS in developing countries. These statements rely on *external* views of the process of OSS development, where authors examine trends in adoption, case studies, and user and developer profiles. In this paper, we take an *internal* view of OSS development and focus on the *essential* properties of OSS. We refer to these properties in the same vein as the classic paper by Fred Brooks, *"No silver bullet – essence and accidents of software engineering"* (Brooks, 1982). Following Brooks, we try to uncover

key properties of OSS projects and to examine how these properties influence public policies for OSS. Therefore, this paper addresses two related points:

- *What are the essential properties of open source?*
- *Taking these properties into consideration, what are the desirable and necessary characteristics of public policies that will promote OSS in developing nations and help its sustainability?*

In this paper, we propose policies for sustainable use of OSS in developing countries. We consider that there are inherent properties of OSS that need to be considered in any public policy that aims at adoption of OSS in developing nations. We set the context in sections 2 and 3, by examining the relation between OSS and knowledge transfer and by examining the profile of OSS practitioners in a large developing nation (Brazil). In section 4, we develop a typology of OSS, based on essential characteristics of software as a technology. Then, in section 5, we consider how these properties bear on public policies for OSS adoption in developing nations. Finally, in section 6, we present a case study of a government policy for an endogenously driven OSS project in Brazil.

2. KNOWLEDGE AND SUSTAINABILITY IN DEVELOPING NATIONS

There is a consensus that knowledge is decisive for development (Reed, 2000). Information, learning, and adaptation are important for sustainability of economies, as much as increasing physical capital. Nevertheless, creating, gaining, and using scientific and technological knowledge in developing countries is a Sisyphean task (Sagasti, 2004). One of the needs for the sustainability of technological projects in developing countries is that new technologies respect and preserve indigenous knowledge and techniques. These technologies have to be well chosen if they are to serve the goals of social and human development of these countries. Otherwise, the new knowledge will only increase the already alarming levels of exclusion and inequality (Reed, 2000).

A key ingredient in setting up ICTs in developing countries is what Braa *et al.* (2004) call *sustainability*. "Sustainability is the challenge to make an information system work, in practice, over time, in a local setting. This involves shaping and adapting the

systems to a given context, cultivating local learning processes, and institutionalizing routines of use that persist over time (Braa et al., 2004)". We aim at understanding OSS as a way by which developing countries gain new knowledge. If ICTs are to be successful in developing countries, they have to be sustainable. For this to happen, the developing country must absorb the knowledge embedded in the technology.

The spread of technology throughout the world encompasses diffusion, absorption and reinterpretation of the new knowledge. "Spread of technology involves interaction between the imported scientific knowledge and the traditional modes of speculative thought" (Sagasti, 2004 p.2). We see OSS as a unique opportunity to leverage the developmental goals of developing countries. OSS should be used to gain knowledge about the technology itself and as a way of creating technology products that fit the specific needs of developing countries. The duality of OSS being at the same time a technology and a product will enable developing countries to take an active role in bringing in ICTs. OSS should be both a way of gaining software development skills and an instrument for social change. To fill this dual role, OSS products should fit the reality of developing countries. These products will trigger social changes when they adequately address the information needs of developing countries.

The success of ICTs in a country is closely related to a national ICT governmental policy (Ehikhamenor, 2002). Government in developing countries has a leading role in the economy and in setting up markets. Nevertheless, the role of OSS for developing countries has to go beyond government mandated use of Linux and other popular open software. OSS has a much more important role in supporting the development goals of these countries. OSS can help developing countries master the technology of software development and enable applications that leverage local knowledge. To reach these benefits, information policies need to rely on a thorough understanding of OSS. We analyze here some principles to guide the design and implementation of strategies to create and gain endogenous science and technology skills in developing countries by using OSS.

3. THE MOTIVATION OF OSS TEAMS: WHAT CHANGES IN DEVELOPING NATIONS?

Several authors have studied the motivations of OSS developers (Amabile, 1996; Deci, Koestner, & Ryan, 1999; Feller & Fitzgerald, 2002; Frey, 1997; Ghosh, Rudiger Glott, Kreiger, & Gregario Robles, 2002; Hertel, Niedner, & Hermann, 2003; Lerner & Tirole, 2002; Raymond, 1999; Rossi, 2004). What emerges from these studies is that OSS developers have different primary motivations, with an emphasis on peer recognition and sense of identification with a community. Lakhani and Wolf (2005) found that "enjoyment-based intrinsic motivation, namely how creative a person feels when working on the project" is more important than what is shown in previous findings which related motivation mainly to external factors in the form of extrinsic benefits. Linus Torvalds thus expresses this sense of identification: "*The act of making Linux available wasn't some agonizing decision that I took from thinking long and hard on it: it was a natural decision within the community that I felt I wanted to be a part of.*" (cited in Rossi, 2004 p.9). However, there are also a growing number of programmers which develop OSS as part of their main job and are paid to do it (Feller & Fitzgerald, 2002). In a survey in Europe, Ghosh et al. (2002) found that most of the OSS developers receive some monetary reward for their work.

There are few comprehensive surveys about the motivation of OSS programmers in developing countries. In Brazil, the Ministry for Science and Technology commissioned a survey on OSS developers which received 3237 responses in which 1953 from developers and 1704 from users (Stefanuto & Salles-Filho, 2005). The survey found that close to 40% of the respondents had a paid job to develop OSS, which is a similar profile found in the European survey reported by Ghosh et al. (2002). Some of the findings of the Brazilian survey include:

- Only 14% of the respondents are involved in OSS product development. The others work mainly on training and customization of existing OSS products;
- A few respondents (20%) are involved in OSS projects that address typical government needs. The most frequent application areas for government

projects are infrastructural, such as Linux-related services (security, network management and web servers).

- Only one-third of the developers share the software they develop, by putting the product on a publicly accessible repository.

The study found there were four interrelated motivational factors for using OSS in Brazil: technical, economic/financial, skill related and ideological. The motivations vary depending on the participant. For corporate users, economic and technical reasons such as *cost reduction*, *greater flexibility to adapt*, *improved quality*, *greater independence of suppliers*, and *greater security* were the main motivating factors. Individual developers were motivated by new skills acquisition and employability. Ideological factors, although present, were of minor importance both for corporate and individual users. The survey also points out the lack of adequate government policies in Brazil to promote sustainable OSS projects. Two key findings support this view. First, only 14% of the developers work in OSS software development, the rest being involved in adaptation of existing products and associated tasks. In addition, only 20% of the developers are working to fulfill typical government needs and those are mostly involved in infrastructure building. These numbers are consistent with the lack of mature OSS products for applications in education, public health, environment, and security (Schmidt & Schnitzer, 2002). They indicate the lack of specific government policies for promoting OSS that addresses the information technology needs of the public sector.

The results of the Brazilian survey support the view that OSS programmers in Brazil share similar motivations to those in developing nations (Feller & Fitzgerald, 2002; Rossi, 2004). This is not surprising, considering that the community of programmers is an internationalized one. Programmers follow the latest international trends and as the Brazilian survey shows, they are not ideologically motivated. They are responsive to professional opportunities, including paid jobs. Thus, it is much more likely that Brazilian programmers engage in OSS projects initiated in developed nations (e.g., by joining a community in *sourceforge*) or get a paid job to adapt an OSS product for a customer. In general, most OSS projects for government organizations are at the add-on or small-scale. A prime example is web services that use OSS tools such as Apache, mySQL and PhP.

The main limit for developing large-scale OSS projects for the Brazilian government and private customers has to do with the impact of legacy software. These are mostly mainframe applications, implemented over a long period. Information technology managers are often unwilling to allow the risk associated to modernizing these products. Programmers associated to these systems are also reluctant to undergo a training program to use OSS tools. Therefore, without a direct public policy for addressing these limitations, OSS use will grow at the fringes of public and private companies and their core applications could remain based on proprietary software.

In short, the profile of OSS developers in Brazil shows a conservative trend. Programmers are usually linked to ongoing projects, both at individual and corporate levels. It is much simpler for an individual to join an existing community than to create a new one. Companies also consider less risky to base their strategies on proven OSS products than to build new ones. This conservative approach does not address the needs of many important IT areas for developing nations. In order to change such a scenario, government should intervene and set up suitable public policies. However, such policies have to consider the specific nature of OSS software. Therefore, we examine the essential properties of OSS in the next section. Based on these essential properties, we argue for appropriate public policies in Section 5.

4. A MODEL FOR OPEN SOURCE SOFTWARE

In this section, we introduce a model to understand OSS based on its essential characteristics. This model will help to us discuss the boundaries of applicability of the OSS model and respond to issues raised by Weber (2004). Based on the model, we also suggest government policies for promoting ICTs in developing countries. Currently, there is an assumption by some that OSS projects are inherently modular and well understood by their developers. This view has its roots in Eric Raymond's manifesto, "The Cathedral and the Bazaar", in which he states the "Linus' law": "*Given enough eyeballs, all bugs are shallow*" (Raymond, 2001). In other words, if a large community of developers understands the source code of the software system, bugs will be discovered at a rapid rate. Along this line, Bollinger et al. (1999) consider that OSS projects should be '*rigorously modular, self-contained and self-explanatory'*.

However, it is important to consider: *are all successful OSS projects modular and self-explanatory?* Modularization and ease of understanding are difficult qualities to achieve at the same time. Decades of experience point out that the most difficult phases of software production are achieving a clear conceptual design (Brooks, 1982) and setting up a feasible strategy for modular development (Parnas, 1972). The two conditions are not easy to achieve simultaneously. As Brooks (1972) points out: *"For efficiency and conceptual integrity, one prefers a few good minds doing design and construction. Yet for large systems one wants a way to bring considerable staff to bear, so the product can make a timely appearance. How can these two needs be reconciled?"* The open source movement has not refuted this overall panorama of software development because proven software engineering principles are also present in OSS (Fitzgerald, 2004). Thus, we argue that OSS has two essential properties. The first property is the *degree of shared conceptualization*¸ that constrains the potential for the software to be understood by a large community of programmers. The second is the *degree of modularity* of the product that constrains the potential for setting up a distributed development team. By assessing how each OSS project fits these two properties, we can build a taxonomy for open source projects. This taxonomy will help setting up information policies to promote the use of open source in developing countries.

4.1. The first essential property: The degree of 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 (Brooks, 1982). The design problem is even more important for OSS. Effective communication between programmers scattered in different places needs sharing 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*'. The two main conditions for shared conceptualization to happen are:

1. The *post-mature* perspective (G. Câmara & Onsrud, 2004): 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 Open Office suite.

2. The *standards-led* perspective (G. Câmara & Onsrud, 2004): 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.

4.2. The second essential property: The degree of modularity

The second property affecting software development is the degree of modularity. A modular software organization enables breaking the project into small pieces and assigning them to different developers. The role of modularity leads to questions such as: what are the limits for modularity in a software product? Are all software products born equal? Are there inherent differences between an operating system, a web server, and a database management system that limit the modularity of each product?

We argue that each different software product has an inherent potential for modularization. All software products have a core part (a *kernel*) and functions that use it (a *periphery*). 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. 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. This claim is consistent with empirical studies that strongly dismiss the idealized conception of open source projects as based on a loose network of developers spread worldwide. Out of more than 400 developers, the top 15 programmers of the Apache web server contributed with 88% of added lines (Mockus, Fielding, & Herbsleb, 2002). Fitzgerald (2004) calls these top programmers 'code gods' and considers that overcoming this problem is one of the challenges of OSS. Sagers (2004) has a more positive attitude towards this ratio (few skilled to many unskilled

programmers). He thinks that restricted access to main parts of the code improves coordination, which in turn affects positively the success of a software project.

One of the principles of the open source movement is the need for modularity. The analysis of the impact of modularity in OSS program team organization has received increased attention from the OSS research community. One of these studies, MacCormack et al. (2004) compare the modularity in the architectures of Linux and in the two versions of Mozilla. The authors use the "dependency structure matrix" (DSM), which expresses dependencies between parts of a complex structure (Sharman & Yassine, 2004). They report that the first public release of Mozilla had a much less modular code than the first public release of Linux. The authors argue the lack of modularity in this version of Mozilla prevented a large community of developers from engaging in the project. This caused Netscape to rewrite the Mozilla kernel. MacCormack et al. (2004) provide evidence that the modularity of the rewritten Mozilla kernel is comparable to that of Linux. Other studies point out the increased acceptance of new Mozilla kernel by the OSS community (Mockus et al., 2002; Reis & Fortes, 2002).

4.3. A Structural Perspective on OSS Projects

Using the essential properties described in the previous sections, we understand better the limits of open source software as a means of producing technology. In what follows, we present a typology of OSS that helps policy makers in setting up information policies to promote its use. We present our model in Figure 1, where we recognize four types of open source software projects varying from low to high potential for shared conceptualization and from low to high potential for modularity.

The four types of OSS development are:

- High shared conceptualization, high modularity (the High-High case);
- High shared conceptualization, low modularity (the High-Low case);
- Low shared conceptualization, high modularity (the Low-High case);
- Low shared conceptualization, low modularity (the Low-Low case).

4.3.1. High shared conceptualization, high modularity

Here we find the prototypical open source projects, those that fit the Linux model. Many of the developers will have a separate job, and do their work in their "spare" time, or in time assigned in agreement with their employer. We call them *community-led* projects. The "high-high" case usually comes up when a software project has a stable design and when it is structurally possible to break it in many independent modules that are suitable for large-scale team development (Narduzzo & Rossi, 2005). In many cases, the design originates from an established standard. This is the case of Linux, where developers had a stable design standard as a basis for the project (the POSIX standard). A simple and efficient kernel allowed the concurrent development of drivers for external items such as hardware devices. The close kinship of Linux to other UNIX flavors, such as BSD, allowed the easy conversion of a whole suite of applications, such as BIND, sendmail, and the GNU software tools (Oram & Loukides, 1995). However, there are strong limits to large-scale modularity in most software projects. In his classic book, *The Mythical Man-Month*, Frederick Brooks (1972) stated his famous law: *"Adding people to a late software project just makes it later"*. His chief argument was the added costs of communication between any new software developer and the group he joins. Therefore, for the communication costs to be minimal, the design has to minimize communication overhead among group members, The ideal situation is to have a careful module design, which may be hard to achieve in practice. As Brooks states in another classic work, "No Silver Bullet" (Brooks, 1982), software design is hard because the state space of a medium-scale software project is much larger than the human capacity to model it. To sum up, the "high-high" case is difficult to achieve. Indeed, it would be counterproductive if all open source projects would fit into this category, since there would be little innovation coming out of the open source movement. Innovation would be limited to reverse-engineering existing designs or following accepted standards.

4.3.2. High shared conceptualization, low modularity

Here we find many projects, including databases, office automation tools, and web servers. There is a large presence of private companies, which aim at entering the marketplace with products similar to the commercial market leaders. Since users and

developers already know similar products, the effort in designing and using them is reduced. Companies benefit from the reduced risk involved in reverse engineering. There can be outside collaborators, but the main design decisions take place within the institution and often should also address the commercial objectives of these corporations. We call them *corporation-led* projects.

The "high-low" arises in two cases mentioned above: when a commercial software has a large market share or when a software technology becomes stable enough for standards to appear. When a single commercial product has a large part of the market, as with personal productivity suites, switching costs will prevent a new commercial product from capturing market share, even if sold at smaller prices. In this case, there is a strong incentive for newcomers to license their products as open source. When there is standard, as with the SQL language for relational database management systems, the design effort is reduced for the developer and the switching costs are minimized for the user. In both cases, developing an open source product may be part of a private company's business strategy and not a community-led effort. Examples include the mySQL database management system, the Open Office suite and the GNOME user interface from Ximian Corporation (Wu & Lin, 2001).

4.3.3. Low shared conceptualization, high modularity

These are projects with a high-degree of innovation (usually there is no commercial counterpart) and that share a relatively simple software kernel. In this case, the innovation takes place at the periphery. Given a stable kernel, programmers can add new modules that need not be understood by all the community. These products often originate in academic environments by researchers and graduate students. We call them *academic-led* projects.

The "low-high" case occurs when a network of developers produces innovative software collaboratively. This case arises from a combination of causes: a technical community which has consolidated links (they may meet regularly at scientific conferences for instance), a stable knowledge domain, and a product whose design allows scalability. One prime example is the R suite of statistical tools (Ihaka & Gentleman, 1996). The basis for this software is the commercial product S-Plus (Chambers, 1998),

whose elegant and simple design enabled the statistical community to design the R suite tools based on the same basic commands as S-Plus. Based on a stable, well-documented design, the statistical community has extended the basic R functionality into a large set of tools. Other examples on this quadrant include the GRASS GIS (Neteler & Mitasova, 2004) suite of programs.

4.3.4. Low shared conceptualization, low modularity

These projects are usually developed by small teams under a public R&D contract. They target a niche application and address specific requirements, or aim to demonstrate novel scientific work. They have a high mortality rate, since most of them have the lifetime of a research grant. We call them *innovation-led* products.

The "low-low" case arises usually from two sources. The first source is a project started by an individual or small group that is not able to attract the interest of the community. A survey of the *sourceforge* OSS repository found a heavily skewed distribution of the impact of OSS projects. Half of the active projects in *sourceforge* have between 0 and 70 downloads and the other half have between 70 and 600,000 downloads (Hunt & Johnson, 2002). The second source is more interesting. Many OSS projects originate from research projects focused on innovation. The open source license is the natural way for sharing a software prototype produced by a research institution. These products are mostly prototypes showing the feasibility of a new design and are not created for commercial use, often lacking end user tools such as adequate documentation. To take them to the marketplace, their innovative features need a large investment in issues such as documentation and reliability, which is beyond the original developers' capacities and interests. Maintaining and supporting an open source software project needs considerable resources, beyond the reach of most academic research groups. Thus, usually it is difficult for a research team to carry out long-term open source projects. Often, for a research prototype to evolve into an open-source product, some of the original developers move from the original research team to a private company. Alternatively, they set up a nonprofit foundation for product support and maintenance. It is unlikely that an open source project that stays in the "low-low" will survive in the long run. Therefore, although many open source projects may start on the "low-low" quadrant,

they must migrate to other quadrants to survive. Migration to the "high-low" quadrant occurs often when a commercial company decides to use a market strategy based on open source licensing, and takes over the development, as discussed above. Migration to the "low-high" quadrant depends on other conditions. The software needs a stable and well-documented kernel, and a core team that controls its evolution. The product also should have enough innovation to attract a large community.

A software project, in its lifetime, may migrate between these categories. An *innovation-led* product might evolve to a *corporation-led* one by incorporating characteristics of market products. Such is the case of the PostgreSQL database management system, which stems from a Berkeley research project (Stonebraker & Rowe, 1986) with added support for the SQL standard and market needs. A *corporation-led* software might evolve into a *community-led* one if their original developers make the necessary investments and adjustment in intellectual property rights to make it accessible to a larger community. This is case of the Mozilla browser and associated tools, originally from Netscape (Godfrey & Lee, 2000). The Apache Web server is an example of an innovation-led project that evolved to a community-led one. A team of programmers decided to take the source code of the National Center for Supercomputing Applications Web server, update it, and release it to the public. It was later renamed "Apache" because of the many patches needed by the original NCSA software (Mockus et al., 2002). One of the interesting consequences of the typology for OSS is that it provides a way to assess the sustainability of projects. This would allow policy makers to take a more active standpoint in supporting open source.

5. PUBLIC POLICY IMPLICATIONS OF THE STRUCTURAL MODEL FOR OSS

5.1. How essential properties of OSS affect public policy

The preceding sections have examined the nature of open source software development and outlined the main characteristics of its production. The implications for developing nations are significant. Many developing nations are currently actively considering policies to support or enforce adoption of OSS by public institutions (Dravis, 2002). The

arguments in favor of OSS adoption by public institutions include (Ghosh, Krieger, Glott, & Robles, 2002):

- *Lower cost:* adoption of personal computers based on OSS for public use can reduce early entry cost by as much as 50%;
- *Independence from proprietary technology*: many governments are increasingly concerned with over-dependence of their markets on a few foreign companies;
- *Availability of efficient and low-cost software:* the virtuous examples of some products (such as Linux and Apache) have encouraged statements about the widespread availability of OSS software for public use;
- *Capacity to develop custom applications and to redistribute the improved products*. Given the "open" nature of OSS, skilled local programmers could adapt the software to fit local needs, and thus increase the efficiency of the services provided by the improved products.

While we consider that there is enough empirical evidence to support "lower cost" and "independence" claims, the assumptions of "software availability" and "ease of customization" are far more problematic and need a closer examination. Most successful open source software tools are infrastructural products, such as operating systems, programming languages, and Web servers. By contrast, the number of mature OSS products that support end user applications is much smaller (Schmidt & Schnitzer, 2002). Operating systems, compilers, and Web servers are the domain of technically qualified IT professionals that have a good knowledge of the English language. By contrast, there is a huge demand by developing countries for applications that address their *information needs*. Addressing these information needs requires IT professionals who understand users' needs and know how to communicate with real users. Thus, the example of Linux is not reproducible in all situations. In developing countries there will be plenty of situations with a low shared conceptualization and low modularity. This is the case with applications in education, public health, environment, and security.

The issue of social production of technology needs also to be addressed, especially for developing nations. The naïve view of open source products considers only

the software development process, with limited regard for its use. Many open source developers take the view that since their product is superior or equivalent to a commercial one, potential users will automatically adopt it. In reality, the development and user communities are different and most users have limited technical knowledge. Concerns such as documentation, local support, training material and best-case examples dictate user choice. In developing nations, language barriers are an added limiting factor. As a result, the effort needed to place open source software in the hands of users worldwide often falls outside the means of committed programmers. In short, the naïve claims in favor of OSS adoption by developing nations often ignore that these products need a large local investment. The investment to adapt these products for local users varies with each product, as is discussed below.

5.2. OSS Structural Constraints and Project Sustainability

The essential properties of OSS have important consequences for information policies in developing countries. In order to these countries benefit from OSS development, their public policies must be different for each of the four cases ("high-high", "high-low", "low-high", "low-low"). In this discussion, we define *sustainability* as the capacity of a software project to adapt and survive to major changes in its current team and in the financial support structure.

5.2.1. Dealing with the High-High Case

This is the simplest case, since products in this range usually have a large community of developers, which are able to endure major changes in team organization. It is conceivable that, in the unlikely event that Linus Torvalds would resign from his role as the chief programmer of Linux, there would be qualified replacements for the job. Therefore, developing countries are safe to assume that adoption of "high-high" OSS is a safe and sustainable choice. When adopting "high-high" OSS, the main concern for developing countries is one of adaptability. Developing nations need to invest in capacity building, documentation and user training to increase the chances of success of "high-high" OSS adoption.

5.2.2. Dealing with the High-Low Case

This case presents a large challenge to developing nations and policy makers worldwide. Many OSS products in this category are associated with private companies. The programmers have a full-time job as software developers for a company, which in turn will be dependent on revenues associated with services it might provide. Two examples are the mySQL relational DBMS and the Qt user interface toolkit, both products of private companies. This is a case where the open source credo is not fully applicable, since the OSS users may become as dependent on a private company as with proprietary software. Should that company's business strategy fail and the project be abandoned, its users would be in trouble. If possible, developing nations should be careful when adopting "high-low" software products whose long-term sustainability is doubtful, especially if these products are strongly associated to private companies. Adoption of corporation-led software should be preceded by an analysis of alternatives, and when possible, "high-high" products should be considered preferable to "high-low" products.

In some areas, there are few current alternatives to "high-low" software, as in the case of the Open Office suite. In this case, it is important to address the question of governance models associated to such products. Many authors consider that the governance model of an OSS product is just as important as the product itself (Franck & Jungwirth, 2002). There has been an increasing emphasis on governance models that increase the power of stakeholders in the software control and reduce the main developers' capacity for independent decision. In this case, by actively taking part in as stakeholders in such governance boards, developing nations could reduce their liabilities when adopting "high-low" software produced by private companies.

5.2.3. Dealing with the Low-High Case

The low-high case represents a favorable condition, since the modularity of the software design and the existence of an established community indicates that software projects in this area will be sustainable. Since most of the developments in this area are extensions of the kernel, the product grows without major risks. The main challenge here for developing nations is the expertise needed in using these software products, since they

contain a fair amount of innovation. For example, to benefit from the set of applications available in the R suite of statistical tools, users in developing nations need to be technically skillful in advanced statistics techniques. Policy makers in the developing world should be aware of the need for significant investments in human resources, if the "low-high" OSS products are to make a significant impact in their nations.

5.2.4. Dealing with the Low-Low Case

The low-low case affects developing nations in two different contexts. First, users in developing nations may be tempted to adopt products in this category, originally from researchers in the developed world. Since "low-low" projects are unlikely to be sustainable, their adoption entails a significant risk. Before adopting such software, policy makers must assess the likelihood that these projects migrate to the "low-high" or the "high-low" quadrants. If enough resources are available in a developing nation, a team of skilled local programmers could envisage undertaking the task of creating a stable product from a research prototype.

A second possibility is the case of projects started in developing nations. These projects are mostly financed by government grants, associated to local research groups. Unaware of the structural characteristics of OSS products, policy makers might naively believe that, after a initial incentive, an OSS product will blossom by itself. Often, after a initial one-to-three year grant, the project might die out, without attracting a large enough community (or a commercial company) that would ensure long-term sustainability. Policy makers in developing nations should ensure that locally developed products have enough support and guidance to leave the "low-low" situation and migrate to a more sustainable position.

5.3. Appropriate OSS Policies for Developing Countries

In a recent interview, Linus Torvalds stated: "*I think that if the developing country is serious about not just seeing ICT as a cost center, but as a requirement for national development, the real advantage of open source ends up being able to build up your own knowledge base. And that is not cheap in itself – you'll likely pay as much for that as you'd pay for a proprietary software solution. The difference being that with the*

proprietary solution, you'll never catch up, and you'll have to pay forever, without ever learning anything yourself" (cited in Weerawarana & Weeratunga, 2004 p.86).

The point Torvalds made is important for understanding the rôle that governments must play in developing countries. First, government has a strong buying power that can drive the market. Second, state sponsored universities are the source of qualified engineers and their most important source of research funds is the government. Third, as Wilson (2004) argues in his analysis of the struggle of developing countries to follow the information revolution, political institutions and nationwide policies are as important as technology.

The role of OSS for developing countries cannot be restricted to government mandated use of Linux, as has been reported recently (Rossi, 2004). For instance, the Brazilian government is recommending that its agencies have Linux installed in all new computers from 2004 on. In Thailand, the government is aiming at having 5% of its computers running Linux. Nevertheless, OSS has a much more important role. OSS may help developing countries master the technology of software development and support applications that leverage local knowledge. Therefore, development policies should address these broader aspects of OSS.

For instance, Sagasti (2004) suggests some principles to guide implementation and acquisition of science and technology in developing countries. He says that "strategies and policies for establishing an endogenous science and technology base must be fully incorporated into the design of a comprehensive development strategy for the country" (p.85). Isolated technology projects have less chance to succeed or at least to be sustainable in the long run. Since OSS is a technology from which tangible benefits can be harvested early, its integration on long-range policies is more likely to happen. OSS can be used to build products that will give a large portion of the population access to services that it would not have otherwise. These kinds of products are likely to have a positive impact on the public opinion making it easier for government to include support for OSS in its developmental policies.

Another principle suggested by Sagasti (2004) is that "the cumulative process of building endogenous science and technology capabilities requires continuous and sustained efforts over a long time" (p.86). Is OSS sustainable as a long-range

development strategy for developing countries? For development projects to be sustainable it is necessary to incorporate indigenous knowledge and techniques. Although OSS has a great potential for doing this, it also needs support by having government-funded research and training.

An analysis of the challenges facing OSS (Fitzgerald, 2004) suggests other directions for policies in developing countries. Fitzgerald mentions the key role of project leaders. These are individuals with leadership and programming skills. For instance, considering Habermas' (1971) three categories of possible knowledge: technical, practical, and emancipatory. Emancipatory knowledge is achieved by combining the two other types of knowledge. We argue that emancipatory knowledge will be gained by developing countries with an adequate use of OSS. The role of leaders is fundamental. Information policies need to address this important point providing for selection, training, and support of leaders that will help bring together two kinds of knowledge, technical and practical. Project leaders will embody emancipatory knowledge. Leaders will help to spread technical knowledge and will make sure that local knowledge is embedded in the products of software development.

There is a dual role for OSS in developing countries. Government policies need to address both OSS as a technology and as a final product. The example of Linux as a high-high product is not easily reproducible. In developing countries, there will be plenty of projects with a low-low profile. This is the case with applications in education, public health, environment, and security. Often, these applications do not have satisfactory OSS solutions currently available. Also, there are inherent market failures and cultural issues in open source software production, which restrict the chances of success of products from developing nations. Therefore, if governments in developing nations aim to profit from the potential benefits of open source, they must intervene and dedicate public funds to support the establishment and long-term maintenance of open source software projects. The next section presents one example of such government action.

6. A CASE STUDY OF AN OPEN SOURCE GIS PROJECT IN BRAZIL

In this section, we present a case study of a government-funded project for developing OSS in geographical information systems (GIS) in Brazil. GIS is an application area that has a large potential impact on public policy. These systems are used by public agencies to manage urban areas and for environmental monitoring. The worldwide market for GIS software was estimated to be US$ 1,5 billion in 2003 (Daratech, 2003). The potential benefits of adopting open source GIS in developing nations are substantive. Consider, for example, the case of urban cadastral systems based on GIS technology for middle-sized cities. The typical base cost of a commercial spatial database solution for one city is US$ 100,000. Should 10 cities adopt such solution in a given year, there is a saving of US$ 1 million each year on licensing fees, which can finance local development and local adaptation. There is also an extra benefit of investing on qualified staff.

Since 2000, the Brazilian government has been funding a large-scale open source GIS project. The project is *TerraLib*, an open-source library for GIS and associated applications (Gilberto Câmara et al., 2000). TerraLib enables quick development of GIS applications and is available at www.terralib.org. As a research tool, TerraLib aims to enable GIS prototypes that would include recent advances in GIScience. On a practical side, TerraLib supports custom-built applications using spatial databases. The main driving forces behind the TerraLib are the National Institute for Space Research (INPE) and the Catholic University of Rio de Janeiro (PUC-RIO). INPE has a mission to develop science, technology and applications for space-related fields. PUC-RIO is home to one of Brazil's leading research groups in Computer Science. The TerraLib project came out of the need to offer Brazilian users an alternative to commercial GIS software. The software is not a clone of any commercial product, and aims to offer functionalities for spatio-temporal data handling that are not available in any commercial or open source GIS software. Starting in 2001, INPE and PUC-RIO invested more than 50 person-years of programming effort in TerraLib.

It is useful to consider the TerraLib project on the light of the OSS typology proposed in this paper and its public policy implications. The TerraLib project started as

a research initiative to provide an innovative environment for GIS applications being thus located in the "low-low" quadrant. The research qualities of the project were sufficient to get support from Brazilian research agencies. However, when the institutions involved considered that the project was mature enough for a production release, they devised a strategy to move the project to a more sustainable situation. INPE and PUC-RIO considered two alternatives to take the TerraLib project out of the "low-low" quadrant. The first was to move the project into the "low-high" quadrant (low shared conceptualization, high modularity) and the other was to move the project into the "high-low" quadrant (high shared conceptualization, low modularity).

The transition of TerraLib to the "low-high" quadrant was considered difficult because of the nature of the geoinformation technology. A typical GIS application consists of a core of functions that access a spatial database, and a set of customized user interfaces that fit the user's needs. These user interfaces are difficult to share, since each application (e.g., an urban cadastre in a municipality) has specific requirements. In fact, this customization of a core library of functions is a task carried out by service companies. The kernel of these GIS application is a tightly integrated set of functions that are best maintained by a small team of skilled programmers.

Therefore, INPE and PUC-RIO chose to transition TerraLib to the "high-low" quadrant. The Brazilian government continues to support the core team of developers of the kernel and has provided additional support for building a shared conceptualization of the product. These resources have been assigned mainly for two tasks: capacity building for commercial and public users, and direct support for service companies that use the software. INPE and PUC-RIO have invested heavily in user documentation and direct contact with commercial companies that could use the library for providing value-added services to GIS market. There is evidence that this strategy is paying off. On early 2006, more than 10 private companies in Brazil develop products using TerraLib. The latest Brazilian GIS market survey estimates the total market to be US$ 150 million, with 200 companies and 4,000 employees (Magalhaes & Granemman, 2005). The service provider market is estimated to be US$ 40 million. Companies offering GIS services based on open source software form 10% of the service provider market.

One of the important decisions on the TerraLib project was to decide on its open source license. There is a strong debate on the policy governments should take on publicly funded software. Smith (2002) and Evans (2002) argue that publicly funded software should not be licensed using the GPL (GNU General Public License) (Hahn, 2002). They consider the limits built in the GPL prevent commercial companies from using GPL-ed software to produce innovation and promote economic growth. On the other hand, Lessig (2002) considers that sometimes it makes sense for the government to license publicly funded software by the GPL. Lessig considers that government has broader interests than those of commercial companies, and the GPL helps keeping publicly funded software in the "public commons". In the TerraLib case, the decision considered the properties of the GIS market. The GIS software market is an oligopoly in which two companies (ESRI® and Intergraph®) have a market share of 50% (Daratech, 2003). Therefore, there is a "lock-in" effect (Arthur, 1994) in the users' choice of products. INPE considered there should be a strong incentive for commercial companies to use TerraLib to reduce the "lock-in" effects of the GIS market in Brazil. Therefore, TerraLib was released as open source according to the LGPL (Lesser GNU Public License). The LGPL allows private companies to build their applications on top of OSS, and market them as proprietary software. The impact on the commercial market of TerraLib-based products is an indicator of a decrease on the "lock-in" effect, because of a suitable licensing policy.

7. CONCLUSIONS

This paper examined the question "How can OSS be promoted effectively in developing countries?" We addressed this question proposing a more comprehensive view of the use of OSS by developing countries. We saw OSS as having a dual role. First, OSS will let developing countries learn about information technologies, about the technology itself, and about the process of developing software. Second, developing countries may also be able to learn more about themselves by using the technology. This dual role points out that OSS can help to solve the problem of information needs of developing countries. The dual nature of OSS as discussed in this paper will open the opportunity for it to be used to explore the environment where it is going to be used. The

resulting open systems will be adapted to local conditions and will embed indigenous knowledge. In short, OSS will let developing countries make the bridge between a foreign technology and its application to local conditions. The open nature of OSS will enable developing countries to master the technology. The open architecture of OSS enables participation and will let the countries learn about the real conditions in which the systems need to be applied. The product of the use of OSS is software that embodies local knowledge and is adapted to local conditions.

Nevertheless, OSS will not be used adequately in developing countries if the necessary public policies are not in place. In order to serve as a guide to the creation of effective policies, we also developed a structural analysis of OSS. We focused on two properties that highlight the dual nature of OSS: the degree of shared conceptualization and degree of modularity. OSS projects succeed when many developers understand the conceptual design and when the software architecture is well-designed to enable collaborative work.

The analysis of the combination of different levels (high and low) of the degree of shared conceptualization and the degree of modularity led to different perspectives for policies in developing countries. While a “high-high” case may be a safe and sustainable choice because of the existence of a large community of users having proven models to support their work, a “low-low” case may lead to failure because of the reverse reasons. The intermediate situations (“high-low” and “low-high”) represent different risks for developing countries. The low-shared-conceptualization and high-modularity case represents a favorable condition. New projects are extensions of established projects. The high amount of expertise required to develop and maintain these projects requires policy makers to provide for significant investments in human resources. The opposite case (high-shared-conceptualization and low-modularity) is more challenging. Most OSS products in this category are commercial products. This may lead to a dependency on private companies. A recommended policy in these cases is an emphasis on governance models that increase the power of stakeholders in the software and reduce the main developers’ capacity for independent decision. These policies would reduce liabilities for countries adopting “high-low” OSS produced by private companies.

We have argued the view of OSS as a product of a team of committed individuals is not realistic. Most products are built either by a small team of individuals or by corporations. Large collaborative networked teams are responsible for a small number of OSS products. Additionally, most projects aim at reverse-engineering existing designs or at complying with standards. Given the constraints in open source software production, such advances will not happen spontaneously and will require public intervention to fund innovation. Open source software in developing nations needs strong and wise policies to be successful. It is a combination of institutional vision, qualified personnel and strong links to user community. OSS in developing countries needs to be government-funded to be viable. In this paper we presented a view of OSS that will help the creation government policies to use OSS technology to promote their development goals.

ACKNOWLEDGEMENTS

Gilberto Camara's work is partially funded by CNPq (grants PQ - 300557/19996-5 and 550250/2005-0) and FAPESP (grant 04/11012-0). Frederico Fonseca's work is supported by the National Science Foundation under NSF ITR grant number 0219025. Gilberto Camara would like to thank Harlan Onsrud (University of Maine, USA) for many important discussions on intellectual property and open source software.

REFERENCES

Amabile, T. (1996). *Creativity in context*. Boulder, Colo.: Westview Press.

Arthur, B. (1994). *Increasing Returns and Path Dependence in the Economy*. Ann Arbor, MI: The University of Michigan Press.

Barton, J., Alexander, D., Correa, C., Mashelkar, R., Samuels, G., & Thomas, S. (2002). *Integrating Intellectual Property Rights and Development Policy*. London: UK Department for International Development - Commission on Intellectual Property Rights.

Bollinger, T., Nelson, R., Self, K., & Turnbull, S. (1999). Open source methods: peering through the clutter. *IEEE Software, 16*(4), 8–11.

Braa, J., Monteiro, E., & Sahay, S. (2004). Networks of Action: Sustainable Health Information Systems Across Developing Countries. *MIS Quarterly, 28*(3), 337-362.

Brooks, F. (1972). *The Mythical Man-Month*. Reading, MA: Wesley Publishing Company.

Brooks, F. (1982). No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer, 20*(4), 10-19.

Câmara, G., & Onsrud, H. (2004). Open-Source Geographic Information Systems Software: Myths and Realities. In J. M. Esanu & P. F. Uhlir (Eds.), *Open Access and the Public Domain in Digital Data and Information for Science: Proceedings of an International Symposium* (pp. 127-133): U.S. National Committee for CODATA, National Research Council.

Câmara, G., Souza, R., Pedrosa, B., Vinhas, L., Monteiro, A. M., Paiva, J., et al. (2000). *TerraLib: Technology in Support of GIS Innovation.* Paper presented at the II Brazilian Symposium on Geoinformatics, GeoInfo2000, São Paulo.

Chambers, J. M. (1998). *Programming with Data*. New York, NY: Springer-Verlag.

Daratech. (2003). *GIS Markets and Opportunities 2003 Survey*. Cambridge, MA: Daratech Inc.

Deci, E. L., Koestner, R., & Ryan, R. M. (1999). A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. *Psychological Bulletin, 125*(6), 627-688.

Dravis, P. (2002). *A Survey on Open Source Software*. San Francisco, CA: The Dravis Group.

Ehikhamenor, F. A. (2002). Socio-economic factors in the application of information and communication technologies in Nigerian print media. *Journal of the American Society for Information Science and Technology, 53*(7), 602-611.

Ehikhamenor, F. A. (2003). Information technology in Nigerian banks: the limits of expectations. *Information Technology for Development, 10*(1), 13-24.

Evans, D. S. (2002). Politics and Programming: Government Preferences for Promoting

Open Source Software. In R. W. Hahn (Ed.), *Government Policy toward Open Source Software*. Washington, DC: AEI-Brooking Joint Center for Regulatory Studies.

Feller, J., & Fitzgerald, B. (2002). *Understanding Open Source Software Development*. New York, NY: Addison-Wesley.

Fitzgerald, B. (2004). A Critical Look at Open Source. *IEEE Computer, 37*(7), 92-94.

Franck, E., & Jungwirth, C. (2002). *Reconciling investors and donators - The governance structure of open source.* Zurich: Working Paper No. 8, Chair of Strategic Management and Business Policy, University of Zurich.

Frey, B. S. (1997). *Not just for the money: an economic theory of personal motivation*. Cheltenham, UK; Brookfield, Vt.: Edward Elgar Pub.

Ghosh, R. A., Krieger, B., Glott, R., & Robles, G. (2002). *Open Source Software in the Public Sector: Policy within the European Union*. Maastricht: International Institute of Infonomics, University of Maastricht, The Netherlands.

Ghosh, R. A., Rudiger Glott, Kreiger, B., & Gregario Robles. (2002). *The Free/Libre and Open Source Software Survey and Study—FLOSS Final Report*. Maastricht, The Netherlands: International Institute of Infonomics, University of Maastricht.

Godfrey, M. W., & Lee, E. H. S. (2000). *Secrets from the Monster: Extracting Mozilla's Software Architecture.* Paper presented at the 2nd International Symposium on Constructing Software Engineering Tools.

Habermas, J. (1971). *Knowledge and human interests*. Boston: Beacon Press.

Hahn, R. W. (2002). Government Policy toward Open Source Software: An Overview. In R. W. Hahn (Ed.), *Government Policy toward Open Source Software*. Washington: Brookings Institute.

Hertel, G., Niedner, S., & Hermann, S. (2003). Motivation of software developers in the F/OSS projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy, 327*, 1159-1177.

Hunt, F., & Johnson, P. (2002). *On the Pareto distribution of Sourceforge projects.* Paper presented at the The Open Source Software Development Workshop, Newcastle, UK.

Ihaka, R., & Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics, 5*(3), 299-314.

Lakhani, K., & Wolf, R. G. (2005). Why Hackers Do What They Do: Understanding Motivation Efforts in Open Source Projects. In J. Feller, B. Fitzgerald, S. Hissam & K. R. Lakhani (Eds.), *Perspectives on Free and Open Source Software* (pp. 3-22). Cambridge, MA: MIT Press.

Lerner, J., & Tirole, J. (2002). Some simple economics of F/OSS. *Journal of Industrial Economics, 52*, 197–234.

Lessig, L. (2002). Open Source Baselines: Compared to What? In R. W. Hahn (Ed.), *Government Policy toward Open Source Software*. Washington, DC: AEI-Brooking Joint Center for Regulatory Studies.

MacCormack, A., Rusnak, J., & Baldwin, C. (2004). *Exploring the Structure of Complex Software Designs: An*

Empirical Study of Open Source and Proprietary Code (Harvard Business School Working Paper No. 05-016). Harvard, MASS: Harvard University.

Magalhaes, G., & Granemman, E. (2005). *A Survey of Geospatial Market in Brazil*. São Paulo: GITA Brasil.

Mansell, R., & Montalvo, W. d. (1998). *Knowledge societies: information technology for sustainable development*. New York: Oxford University Press.

Mockus, A., Fielding, R., & Herbsleb, J. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, 11*(3), 309 - 346.

Narduzzo, A., & Rossi, A. (2005). The Role of Modularity in Free/Open Source Software Development. In S. Koch (Ed.), *Free/Open Source Software Development*. Hershey, PA: Idea Group.

Neteler, M., & Mitasova, H. (2004). *Open source GIS: a GRASS GIS approach* (2nd ed.). Boston: Kluwer Academic Publishers.

Oram, A., & Loukides, M. (1995). *Programming with GNU Software*. Sebastopol, CA: O'Reilly.

Parnas, D. L. (1972). On the Criteria to be Used in Decomposing Systems into Modules. *Communications of the ACM, 15*(2), 1053-1058.

Raymond, E. S. (1999). A Brief History of Hackerdom. In C. DiBona, Ockman, S., Stone, M (Ed.), *Voices from the Open Source Revolution*. Sebastopol, CA: O'Reilly & Associates.

Raymond, E. S. (2001). *The cathedral and the bazaar: musings on Linux and Open Source by an accidental revolutionary* (Rev. ed.). Cambridge, Mass.: O'Reilly.

Reed, A. M. (2000). *Rethinking Development as Knowledge: Implications for Human Development*. Rome: United Nations Office for Project Services.

Reis, C., & Fortes, R. (2002). *An Overview of the Software Engineering Process and Tools in the Mozilla Project.* Paper presented at the Workshop on Open Source Software Development, Newcastle UK.

Rossi, M. A. (2004). Decoding the "Free/Open Source (F/OSS) Software Puzzle" a survey of theoretical and empirical contributions. *Quaderni dell'Istituto di Economia, 424*, 1-40.

Sagasti, F. R. (2004). *Knowledge and innovation for development: the Sisyphus challenge of the 21st century*. Cheltenham, UK; Northhampton, MA: E. Elgar.

Sagers, G. W. (2004). *The Influence of Network Governance Factors on success in Open Source Software Development Projects.* Paper presented at the 25th International Conference in Information Systems (ICIS 2004).

Schmidt, K. M., & Schnitzer, M. (2002). *Public Subsidies for Open Source? Some Economic Policy Issues of the Software Market*. Munich: Seminar for Economic Theory, Ludwig Maximilian University.

Sharman, D., & Yassine, A. (2004). Characterizing Complex Product Architectures. *Systems Engineering Journal, 7*(1).

Smith, B. L. (2002). The Future of Software: Enabling the Marketplace to Decide. In R. W. Hahn (Ed.), *Government Policy toward Open Source Software*. Washington, DC: AEI-Brooking Joint Center for Regulatory Studies.

Stefanuto, G. N., & Salles-Filho, S. (2005). *Impact of the free software and open source on the software industry in Brazil*. Campinas, Brazil: UNICAMP, available at http://observatorio.softex.br/components/com_observatorio/arquivos/Softex%20ingles%20para%20site.pdf.

Stonebraker, M., & Rowe, L. A. (1986). The Design of POSTGRES. In *ACM-SIGMOD International Conference on the Management of Data* (pp. 340-355). Washington, D.C.

UN. (2003). *E-Commerce and Development Report* (No. UNCTAD/SIDTE/ECB/2003/1). New York and Geneva: United Nations.

UN. (2004). *Road Maps towards an information society in Latin America and the Caribbean* (No. LC/G.2195/Rev.1-P). Santiago, Chile: United Nations - Economic Commission for Latin America and the Caribbean.

Weber, S. (2004). *The Sucess of Open Source*. Cambridge, MASS: Harvard University Press.

Weerawarana, S., & Weeratunga, J. (2004). *Open Source in Developing Countries* (No. SIDA3460en). Stockholm, Sweden: Sida 2004 - Department for Infrastructure and Economic Cooperation.

Wilson, E. J. (2004). *The information revolution and developing countries*. Cambridge, Mass.: MIT Press.

Wu, M.-W., & Lin, Y.-D. (2001). Open source software development: an overview. *IEEE Computer, 34*(6), 33-38.