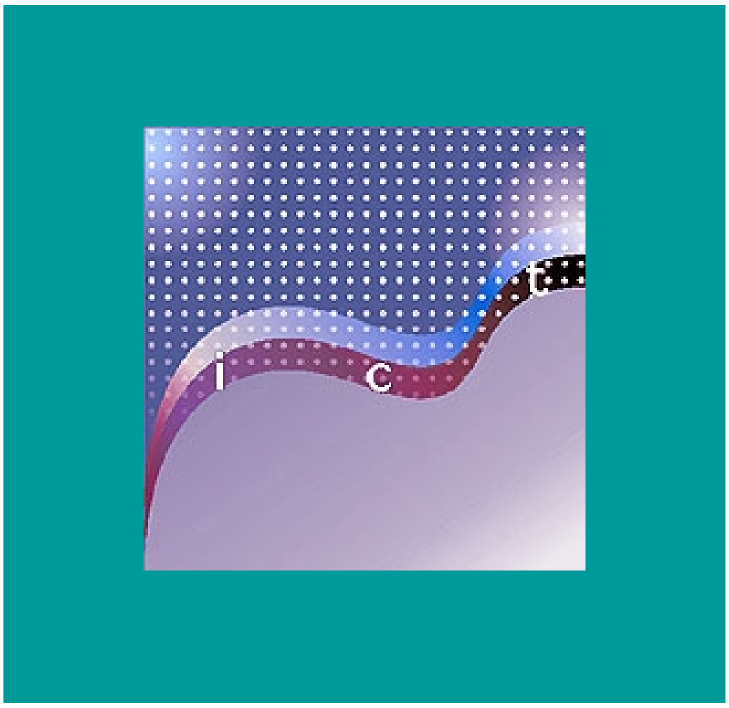
United Nations Conference on Trade and Development

E-COMMERCE AND DEVELOPMENT REPORT 2003

Internet edition prepared by the UNCTAD secretariat

Chapter 4: Free and open-source software:
Implications for ICT policy and development



UNITED NATIONS
New York and Geneva, 2003

UNCTAD/SIDTE/ECB/2003/1

Note

The English version of the full report and the English, French and Spanish versions of its Overview section are currently available on the Internet at the address indicated below. Versions in other languages will be posted as they become available.

http://www.unctad.org/ecommerce/

UNCTAD/SDTE/ECB/2003/1

UNITED NATIONS PUBLICATION

Sales No. E.03.II.D.30

ISBN 92-1-112602-9

Chapter 4

FREE AND OPEN-SOURCE SOFTWARE: IMPLICATIONS FOR ICT POLICY AND DEVELOPMENT

Free and open-source software (FOSS) challenges our preconceptions about how software is produced and distributed. The software industry today generates yearly revenues in excess of $300 billion. FOSS is software that has made its source code public and allows – perhaps even motivates – users to change the source code and redistribute the derivative software. Liberating the source code supports broad collaborative development in software production, better porting[1] with other programmes produced by independent programmers, and the customization of software to meet different commercial, regulatory, cultural and linguistic requirements. Most importantly, in particular for developing countries, FOSS allows today's and tomorrow's experts and information technology (IT) leaders to acquire skills and advance their knowledge rapidly.

Its technological opposite, closed-source or proprietary software, may not support the information and communication technology (ICT) development process as well because it requires a significant upfront investment in license fees for installation and upgrades; it is not always adaptable to local concerns; and its exclusive or even dominant use may not adequately support the local development of the expert knowledge and skills needed to fully embrace the information economy. While proprietary software has its place and role, Governments should consider their policy position on FOSS in the context of their overall agenda and their ambitions of bridging the digital divide and using ICT for increased, improved trade and development.

A. Introduction

The hardware that makes modern computing and communications possible has advanced at an extraordinary rate in the last few decades, and that process is likely to continue. "Moore's law", which is really an observation of a pattern, states that the capability of microprocessors doubles, while their price falls by nearly one-half, every 18 months.[2] This has created an information-processing ecology where computer hardware is much more sophisticated and reliable than software – the instructions that human beings create for it.

However, there is no Moore's law for software. While computing power falls rapidly in price, software that can make use of that computing power becomes more complicated, sometimes more expensive and less reliable, and almost always more difficult to configure and maintain. Yet it is software that constitutes the fundamental rules for information processing, and thus for an information economy and an information society. Massive processing power connected by ever-increasing bandwidth is a skeletal infrastructure. Software determines how information is manipulated, where it flows, to whom and for what reasons.

Developing countries need to define their ICT strategies and make them relevant to the development process. Policy regarding software use has suddenly become an important issue because a new choice has recently become viable: that of FOSS, and with it the promise of information-enabled development. However, Einstein once commented that "sometimes one pays most for the things one gets for nothing", and this thought is relevant to the FOSS debate. The countries and individuals that will profit from FOSS are those that will strive to formulate their policies in an informed manner and that will contribute back to the FOSS knowledge base.

This chapter explains how FOSS and, for comparison, proprietary software are created – not in a technical sense, but in an organizational sense – and why that matters, for developed and particularly for developing economies. It suggests that the FOSS process produces better software that could match the unending improvements in computer hardware. Like any product, software is simply the outcome of a production process that combines human effort, inputs, and capital of some sort in a distinctive way.

The "standard" way of organizing software production has been much like the standard way of building a complex industrial good: a formal division of labour that uses proprietary knowledge, guarded by restrictive intellectual property rights (IPR) and enclosed within a corporate hierarchy, that guides and governs the process. Following this approach, today's software industry has grown into a colossus estimated to generate more than $300 billion in annual earnings.

Table 4.1

Top 10 software companies, ranked by revenue and market capitalization

		Annual revenue	Market capitalization
		(millions of $)	*(millions of $)*
1	Microsoft	31,375	260,000
2	Oracle	9,487	63,400
3	SAP	7,700	32,300
4	Computer Associates	3,083	12,400
5	VERITAS	1,531	10,100
6	Electronic Arts	2,489	9,300
7	Intuit	1,373	9,000
8	Adobe Systems	1,194	8,000
9	Symantec	1,328	6,600
10	PeopleSoft	1,949	4,700
11	Competition to Top 10[3]	8,445	28,582
	Total	**69,954**	**444,400**

Source: UNCTAD estimates based on data from Yahoo Finance (http://finance.yahoo.com) and Financial Times Market Data and Tools (http://www.ft.com).

According to industry analyst IDC, the packaged software industry[4] alone is worth almost $200 billion, while the Gartner Group puts the market for corporate software at nearly $80 billion.[5] Table 4.1 gives some data on the largest proprietary software producers. These figures should be taken with caution, as not all revenues earned come from selling proprietary software licenses. Indeed, consulting and customizing software for clients is an important activity as well. In addition to the listed firms, IT heavyweights such as IBM, Sun Microsystems and EMC, as well as the major personal computer (PC) hardware producers, also generate significant revenues from corporate software services.[6]

But this is not the only way to organize software production. In the last few years, another way of building software, the open-source process, has gained publicity just as the products of this process, such as GNU/Linux, are gaining market share. In fact, open source is not a new process. However, it is fundamentally different from the leading alternative, and the success of FOSS projects demonstrates that complex software can be built, maintained, developed and extended in a non-proprietary setting where many developers work in a highly parallel, relatively unstructured way, often without direct or immediate monetary compensation.

This chapter establishes and builds on the premise that the open-source process is a viable mode of software production that presents a real choice for firms and Governments making ICT decisions, in particular in developing countries. It aims to elucidate the FOSS phenomenon itself and to clarify some of the issues involved in choosing between open-source and proprietary software. It presents some of the parameters and variables that may influence these choices, along with practical examples of the possibilities and consequences of open-source adoption, using examples from industrialized countries and highlighting initiatives in developing countries. Finally, the chapter provides a framework for understanding the policy implications surrounding FOSS, focusing on choices that the public sector should consider, and on reasons that might influence its decisions.

B. The process and the challenge

What is FOSS, and how is it different from proprietary software products sold under conventional intellectual property (IP) regimes? A simple analogy to any popular cola drink can be helpful.[7] A manufacturer sells bottles of cola soda to consumers. Some consumers may choose to read the list of ingredients on the bottle, but that list of ingredients is surprisingly generic. The manufacturer typically has a proprietary "formula" that it does not reveal. The formula is the knowledge that gives guidelines on how to combine the ingredients in particular proportions, and perhaps with some "secret" flavouring mix, to produce something of commercial value. However, the bubbly liquid cannot be reverse-engineered into its constituent parts. You can buy cola soda and you can drink it, but you cannot *understand* it in a way that would empower you to reproduce the drink or improve on it, and to distribute your copied or improved cola drink to the rest of the world. In order to guarantee that no entity rediscovers, reverse-engineers or (by more devious means) acquires the cola formula, the formula is also subject to IP protection.

The basic economics of IPR provides the rationale for organizing cola soda production this way. The core problem of IP is supposed to be about creating incentives for innovators. Patents, copyrights, licensing schemes and other means of restricting knowledge give legal backing to the notion that economic rents are created and that innovators can and should appropriate some proportion of those rents as incentives to innovate. Without IP protection, should a "new and improved" formula be discovered, the person who invents the new formula would have no defensible economic claim to a share of the profits that might be made by selling drinks engineered from the innovation. That person no longer has a financial incentive to innovate in the first place, so the system unravels and improved cola soda is never produced. While the original producer certainly supports and takes advantage of all available IP protection, it is aware that the security of its formula, and consequently its business, lies in its physical protection, and in the entry costs, as well as the manufacturing and distribution costs, for potential competitors. Thus, the manufacturer complicates the recipe, divides up the formula so that certain individuals know only parts of it while no one knows it in entirety, uses a good safe, and strives to establish a monopoly market position.

The production of proprietary computer software is typically organized under a similar regime, with a parallel argument behind it. When purchasing software, for example, people or companies actually buy a right-to-use license. They do not own the software in the sense that they can do with it what they wish. The right-to-use license permits them to use proprietary software on a computer, but only under very specific terms: they cannot reproduce it, modify it, improve it, or redistribute their own version of the software to others. Copyright, licenses, patents and other legal structures provide a layer of legal protection to this regime, but there is an even more fundamental mechanism that stops license holders from doing any of these things. Just as the cola soda producer will not release its formula, a proprietary software producer will not publicize the software's source code.

Programmers *write* software *source code* using a programming language. Computers *run* software in *binary code* format.

The source code is a list of instructions that make up the "recipe" for a particular software application, such as a word processor or a spreadsheet. Software engineers write source code using a particular programming language (like C++ or Fortran) that experts can read and understand, as well as fix and modify. To non-experts, source code looks like a combination of unintelligible language and mathematical and logical expressions.

Before the software can be used on a computer, it needs to be "compiled". *Compiling* is the process of translating the source code into binary code, consisting basically of series of ones and zeros, after which it is saved as a separate file. Only then can the compiled file run on a computer, at which point it is called the *executable binary file* or the *binary*. Most proprietary commercial software is distributed only as executable binary files, which a human cannot "read" and make sense of. Not having access to the source code restricts users' ability to modify the software. Reverse engineering the binary code back into source code is generally not possible either. Thus, selling only the executable binary files is a very effective way for proprietary software producers to control what users can and cannot do with the software they buy.

Proprietary source code is the touchstone of the conventional IP regime for computer software. Proprietary source code is an important reason why the software industry can generate sizable revenues and earnings (see table 4.1). In turn, these companies distribute part of their profits to the programmers who write their code and, in this way, provide incentives for them to innovate.

The open-source process inverts this logic. The essence of FOSS is that the source code is "free". That is, along with the executable binary files that actually run on the computer, the source code is released[8] for anyone and everyone to examine, use or modify it. "Free" in this context means the freedom to run the programme for any purpose, to study how it works and adapt it to one's own needs, to redistribute copies to others, and to improve the programme and share improvements with the community so that all benefit (FSF 1996). It does not necessarily mean that the price is zero, since FOSS can be traded in markets just like any other artifact. Programmers often explain this seeming incongruity in shorthand such as the following: when you hear about "free" software, think "free speech", not "free lunch"; or "software *libre*", not "software *gratis*".[9]

For example, the popular FOSS GNU/Linux distributions are sold on CD-ROM for prices ranging from several dollars (for discs only) to more than $100 (for packages that include manuals and help-line access for a certain period). Often, the executable binary files can be obtained without paying, but this would require users to download the files from the Internet and burn their own installation CD-ROMs. This approach, too, has a definite cost, involving Internet access (preferably broadband), as well as a printer, a CD burner and blank CDs. Whether or not an ICT business can make money with FOSS is a relevant issue and is discussed in section E of this chapter.

Building complex software is a difficult and exacting task because it involves technical and human complexity in both abstract conception and implementation. People use software in an extraordinarily diverse technological and cultural matrix that changes almost continuously. For example, if an auto engineer has to try to envision the range of conditions under which people will try to drive a car, the software engineer is faced with a harder task because much of the technological environment in which a piece of software will be used has not even been produced or distributed at the moment that the software is being written. Aside from hardware advances, changes in operating system and networking environments will influence how we use software designed today. Highways and bridges simply do not change that fast, and they are not configurable by users in the way that software is.

The software production problem leads unavoidably to a division of labour. The primary questions are: What kind of division of labour? How should this be organized? Putting the right numbers of people in the correct positions is also important but is really a secondary problem.

The standard answer to that question is hierarchical organization in the Fordist[10] style. A clear division between design/architecture and implementation, segmentation of tasks into subsystems that are then supposed to "snap" together, and reporting hierarchies with command and control from above – these are all familiar features of industrial organization. An authority assigns tasks, monitors performance, and compensates according to measurable indicators of execution. Controlling the source code becomes a means of controlling the division of labour.

The open-source process approaches this challenge from a different direction. Once the source code is released, the configuration and management of labour becomes a project of the labourers themselves. *The key elements of the open-source process, as an ideal type, are voluntary participation and voluntary selection of tasks.* Anyone can join an open-source project, and anyone can leave at any time. There is no consciously organized or enforced division of labour. It may be that the underlying notion of a division of labour does not fit the open-source process very well. Labour is *distributed*, and it could hardly be otherwise in projects that at any given time may involve a hundred or even thousands of programmers. But it is not really *divided* in the industrial sense of the term. The discussion of the Apache and GNU/Linux structures in section F of this chapter reflects on how individual motivations translate into collective actions.[11]

C. A history of software production

The concept of "free" software is not new. In the 1960s and 1970s, mainframe computers in university computer science departments and in corporate facilities were principally tools for research. The idea of free access to the source code for the installed software was seen as natural and was often taken for granted. The FOSS environment was needed to advance compatibility among different computer systems for which software had to be reengineered to account for different hardware – an often time-consuming and expensive process. Incompatibility clashed with the scientific ethic of sharing and accumulation of knowledge, as well as the practical problem of having to rewrite extensive amounts of code for different types of computers.

In the United States, AT&T's Bell Labs led the way by focusing effort on development of the UNIX operating system and an associated language for developing applications, called simply C, that could run on different and varied hardware.[12] Under the terms of its regulated monopoly deal with the US Department of Justice, AT&T could not engage in commercial computing activities and thus could not sell UNIX for profit. It seemed almost natural to give away the source code to universities and others who the Bell Labs engineers believed could help them improve the software by finding bugs and fixing the source code.[13]

Thus, UNIX software, typically under copyright, nonetheless was in most cases distributed for free along with the source code.

Concrete incentives supported this very casual and informal treatment of copyright. The behaviour made sense to the owner of the copyright, since software was seen at this time not as a profit center itself but principally as a hook to encourage people to buy hardware. Give away better software, and you can sell more computers – so the thinking went. It also made sense for an innovative programmer to freely give ideas to the software's owner. If all or many of these innovations were incorporated into future software releases, computer departments would not have to bother reintegrating improvements piecemeal, but could simply await the next official release.

The logic of free software began to break down in the late 1960s. In 1969 the US Department of Justice filed a massive antitrust suit against IBM, pushing it to unbundle its "solutions" and begin charging separately for software.[14] IBM subsequently began to ship its new mainframes with operating systems that did not distribute source code. In fact, administrators had to sign non-disclosure agreements simply to get an executable copy. This decision represents the birth of the modern commercial software industry. Microsoft was founded in July 1975 as a company that for all intents and purposes simply wrote and sold software. The arrival of the PC in the early 1980s and its rapid widespread distribution onto desktops in the business world reinforced this trend. Software that at one time had been traded freely among programmers was now an extraordinarily valuable and lucrative product in its own right. The development of a separate software industry and business model had a major impact on the programming profession. Many of the best programmers in the United States and elsewhere were hired away into lucrative positions at spin-off software firms.

In reaction to these developments, in 1984 Massachusetts Institute of Technology researcher Richard Stallman founded a project to revive FOSS activities by creating a complete set of FOSS utilities and programming tools.[15] This initiative led to the establishment of the Free Software Foundation (FSF). The FSF exclusively uses the term "free software" to denote software that allows the user to run, copy, distribute, study, change and improve it through access to the source code. The FSF sees copyright as a means of imprisoning information and creating unequal access, usually along the lines of wealth and poverty. To replace traditional copyright, the FSF has developed a standard copyright agreement, the GNU General Public License (GPL), that is often called "copyleft".[16] GPL is designed to deter programmers from "closing" the source code of an FOSS computer programme and stop anyone from bringing it into a proprietary commercial development environment.[17] Box 4.1 discusses the principal position of the FSF in more detail. Section 3 of this chapter discusses the legal details of the GPL (under the heading "Intellectual Property Rights").

Box 4.1

The Free Software Foundation and the General Public License

The central idea of the General Public License (GPL) is to prevent cooperatively developed open/free software source code from being "enclosed" or turned into proprietary, restrictively copyrighted software. The GPL states that users are permitted to run the programme, copy the programme, modify the programme through its source code, and distribute modified versions to others. What they may not do is add restrictions of their own. This is the "viral clause" of GPL. It compels anyone releasing derivative software that incorporates code "copylefted" under GPL to use the GPL in the new release as well. The Free Software Foundation states: "You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program [any programme covered by this license] or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this license" (FSF 1991).[18]

Stallman and the FSF created some of the most widely used pieces of UNIX software, including the Emacs text editor,[19] the GCC compiler,[20] and the GDB debugger.[21] As these popular programmes were adapted to run on almost every version of UNIX, their availability and efficiency helped to cement UNIX as the operating system of choice for "free software" advocates and leading academic and research institutions. But the success of the FSF was in some sense self-limiting because of the viral nature of the GPL. Its principal position against proprietary software clashed with the utilitarian view of many programmers, who wanted to use pieces of proprietary code together with free code when it made sense to do that, simply because the proprietary code was technically good. The GPL does not permit this kind of flexibility and sometimes poses difficult constraints to developers looking for pragmatic solutions to problems.

The FOSS process depends heavily on communications tools to enable modification, innovation and evolution with regard to the code, with collaboration between diverse and remote localities. While the ARPANET was barely sufficient, the rapid spread of the Internet in the early 1990s accelerated FOSS activity. The development of the GNU/Linux PC operating system software began during this period from honourably modest roots.

In late 1990, Linus Torvalds, a 21-year-old computer science student at the University of Helsinki, started building the kernel of a UNIX-like operating system on his home PC. In autumn 1991, Torvalds released the source code for the kernel of his new operating system named GNU/Linux to an Internet newsgroup, along with a note asking for comments and collaborators. The response was unexpectedly good. By the end of the year, nearly 100 people worldwide had joined the GNU/Linux newsgroup; many of these people were active developers who contributed bug fixes, code improvements and new features. Through 1992 and 1993 the community of developers grew at a gradual pace. This was happening at a time when it was becoming generally accepted within the broader software community that the era of UNIX-based operating systems was coming to an end in the wake of Microsoft's increasingly dominant position (Raymond 2000). In 1994 Torvalds released the official GNU/Linux version 1.0.

While various proprietary versions of UNIX lost market share during the mid-1990s, GNU/Linux steadily acquired market share in the late 1990s and has now become the only credible competitor to Microsoft in the PC operating system market. The growth of GNU/Linux had several causes. Many in the ICT community found the manner in which proprietary software companies leveraged their IP (source code) galling. Others asserted that that the technical quality of proprietary software was suffering from the corporate-style development process. It was claimed that, regardless of how powerful the proprietary software companies grew, they simply could not employ enough testers, designers and developers to thoroughly debug their own software. At the same time, proprietary software firms invited only limited interaction between advanced users and programmers to repair and improve a piece of software.

While GNU/Linux grew, the viral nature of the GPL, as well as the rigour of the FSF's position, gave rise in the mid-1990s to an alternative institution for "free" software, the Open Source Initiative (OSI). The OSI came into being in February 1998 during a meeting of several influential IT experts convened in response to the decision by Netscape to publicize the source code of its browser. Netscape's decision was seen as a lead to follow in promoting the development of FOSS, in particular vis-à-vis the business and corporate community. Instead of including a prescribed copyright or "copyleft" message, the OSI requires entities distributing FOSS to satisfy the Open Source Definition (OSD) in its copyright statement.[22] While the GPL *requires* any redistribution of GPL software to be released only under GPL (to prevent the "closing" of the code), the OSD *allows* redistribution under the same terms, but does not require it. Certain licenses that fall under the OSD entitle a programmer to modify the software and release the modified version under new terms that include making it proprietary. Box 4.2 gives an overview of the OSD.

The OSI emphasized economic competitiveness and aimed its message directly at the mainstream corporate world.[23] The argument was that the open-source process emphasized high reliability, low cost, and better features. Most importantly, a business or Government using FOSS could avoid becoming locked into using software produced by a controlling monopolist. Open-source users would gain autonomy through control of their information systems, which were increasingly the core asset of almost any business. The OSI initially aimed its message at the CEOs of the largest multinational companies and emphasized the various ways in which IT companies themselves could generate economic profits while freeing source code. For example, better software would allow hardware manufacturers to sell more computers and other devices. Customization services that built packages of open-source solutions, then optimized, maintained and serviced them for particular business and government settings, would be extremely valuable.

The corporate world's response was immediate. In January 1998 Netscape announced that it would release the source code for its World Wide Web browser as open-source code. By the summer Oracle and Informix, two of the largest independent software vendors for corporate applications and databases, announced that they would port their applications to GNU/Linux. Over the next several months, other first-tier independent software vendors, including Sybase and the German SAP, made similar announcements. In the first half of 1999 IBM began focusing on GNU/Linux as an operating system for its servers (Berinato 1999, 2000). IBM also became a major supporter of Beowulf supercomputing (CNET 2000). Major US hardware vendors (Compaq, Dell, Hewlett Packard, Silicon Graphics) as well as chip

BOX 4.2

Open source defined

The Open Source Definition (OSD) maintains the following position:

- Source code must be distributed with the software or otherwise made available for no more than the cost of distribution.
- Anyone may redistribute the software for free, without owing royalties or licensing fees to the author.
- Anyone may modify the software or derive other software from it and then distribute the modified software under the same terms.

OSI removes the viral impact of the GPL. Open source does not just mean access to the source code. The OSI "approves" existing licenses as compliant with the OSD. (A recent count found 21 of these, including the GPL license but also licenses from IT corporate heavyweights such as IBM, Nokia and Intel.) The OSI aims to bring pragmatism into the development of technically sophisticated software and discards the FSF ideology. Not everyone shares this goal or sees it as a progressive change. However, it is worth remembering that the philosophical core of the OSI was in a very different place. One of its founders, Eric Raymond, explained:

"It seemed clear to us in retrospect that the term 'free software' had done our movement tremendous damage over the years. Part of this stemmed from the well-known 'free-speech/free-beer' ambiguity. Most of it came from something worse – the strong association of the term 'free software' with hostility to intellectual property rights, [with] communism, and [with] other ideas hardly likely to endear themselves to an MIS manager" (1999a).

manufacturers Intel and AMD have all made major commitments to GNU/Linux. For-profit companies that provide auxiliary services and support for GNU/Linux, such as Red Hat Software in the United States, SuSe in Germany and MandrakeSoft in France, started commercial operations in the late 1990s. Apache continued to increase its lead in the Web server market just as the Web itself was exploding in popularity. In October 2000, Sun Microsystems released the source code for StarOffice, a software suite for everyday office use, in order to establish OpenOffice.org. These and other important or popular FOSS applications are described in box 4.3.

Microsoft began to see the open-source process in general, and GNU/Linux in particular, as a major credible threat to the market presence of its Windows operating systems, on servers and perhaps even on desktop PCs.[24] A high-level internal Microsoft memorandum issued in 1998 was leaked on 31 October and became known as "the Halloween Memo". It reportedly portrayed FOSS as a direct short-term threat to revenues and dominant position in some markets. It was also a long-term strategic issue because, according to the memo, "the intrinsic parallelism and free idea exchange in OSS [FOSS] has benefits that are not replicable with our current [proprietary] licensing model."[25]

The brief history of software, from its open-source roots to proprietary models and, now, the journey back to open source, has appeared to take place mainly in the United States. Indeed, Lancashire (2001) supports this notion and gives some geographic data for participating developers. In a sense, this phenomenon is self-explanatory, as the majority of FOSS developers will be based in the countries with the most developed software industries. A look back at table 4.1 reveals that, of the top 10 global software firms and their 10 main competitors, only three are not based in the United States and only one is in a developing country.[26] However, the situation seems to be rapidly changing, and, judging from the results of the survey reported in section H of this chapter, FOSS activities in developing countries may become increasingly visible in the near future.

By the end of the 1990s the FOSS process had proved its viability as a means for building complex software packages that could compete successfully with proprietary products, and in an increasing number of IT market segments, from low-end embedded processing applications to grid-based supercomputing. Companies as diverse as GNU/Linux distributor Red Hat and traditional IT giant IBM have learned how to generate sustained profits by providing services using various kinds of FOSS. It is now clear that there are at least two discrete models for organizing the production of software. Both appear to be sustainable. Today Governments, businesses and almost anyone who uses software can make choices, and will need to make choices, between and among products generated through both processes of building software.

D. Is FOSS better?

The ultimate issue that the open-source process has to contend with is achieving quality equal to or superior to that achieved in proprietary corporate organizations. It can do so in four ways.

1. While affirming that while all software has programming errors (bugs) and stability problems, FOSS can have more developers looking critically at problems and proposing fixes than any proprietary software corporation. In other words, "given enough eyeballs, all bugs are shallow" (Raymond 2000).

2. Because FOSS is not hampered by the marketing and business dynamics of maximizing revenue from license sales, developers can and do release bug fixes, patches and new versions more frequently.

3. The installation of proprietary software following the purchase of a right-to-use license is often tied to accepting terms and conditions that decline any liability for damages resulting from its use, beyond the replacement of the hard disk drive where the software was installed – clearly not a hard guarantee of quality with which to compete.

4. Source code availability is in itself an important product quality. Imagine a transportation company purchasing a fleet of vehicles that arrive without the keys to the engine hoods; the keys would be useless because the company has agreed in a contract with the manufacturer that it will in no way attempt to fix or inspect the vehicles' engines. Such vehicles, like analogous software, are obviously inferior.

The FOSS process is, however, neither fool- nor failure-proof. One possible problem is the fragmentation and forking of projects: a collaborative team may come to loggerheads over technical issues, or even personality problems. Fragmentation, or forking, means that existing development resources are split between the main and dissenting teams and users may be faced with unwanted choices and compatibility issues. Another cited problem is that it is difficult for users to clearly predict where the development may be going in terms of future versions, functionalities or hardware support. Finally, developers and project team leaders may simply lose interest or reestablish themselves in a way no longer relevant to the software project. However, these problems are not the exclusive territory of FOSS. Proprietary software carries its heavy share of differing standards and compatibility. Often, good software has been produced by companies that did not achieve comparable financial results, thus forcing consumers to switch to products from better-managed companies. Software support for new hardware in the proprietary software world is often conditional on a forced "choice" to upgrade and pay anew for licenses.

No single software can be unambiguously "better" than all others. Like any tool, software has certain characteristics of usability, reliability, flexibility, robustness and cost. There is no single optimal balance between these characteristics, and much depends on the distinctive needs of a particular user. All things being equal, however, software with fewer serious bugs and a lower total cost of ownership is generally preferable on simple economic grounds. Yet even these criteria are hard to measure. An often-used test of robustness is the average uptime. Table 4.2 gives an overview of Web servers with the longest uptime measured during the week of August 18, 2003, and the operating system and server software that they use. It is notable that only one of the 20 most robust Internet servers runs on proprietary software.

Because bugs appear while software is used in diverse environments, there may not be precise or reliable means of estimating the scope or seriousness of a particular programme's bugginess. More important is how quickly a bug, once identified, can be fixed. A recent study compared bug resolution for three matched pairs of FOSS and closed-source proprietary programmes: two Web servers, two operating systems, and two graphical user interface (GUI) packages. It found some support for the hypothesis that open-source bug reports are resolved faster than closed-source service requests, after controlling in some ways for the priority and severity of each request.[27] This is a very cautious finding, since it may be that bugs are uncovered at different rates as well, and have different characteristics of complexity across types of software. The result is consistent with the expectation that users are most highly motivated to fix what gets in the way of their intended use if they are empowered to do so by having access to source code.

Calculations of total costs of ownership (TCO) try to capture fully the costs of deploying, maintaining and using a system over the course of its lifespan. Studies

Table 4.2

Web servers with longest average uptime

		Average uptime	Operating	
Rank	**Site**	**(days)***	**system**	**Server software**
1	www.daiko-lab.co.jp	1569	FreeBSD	Apache/1.2.4
2	www.rfj.ac.se	1389	BSD/OS	Apache/1.3.26 (Unix)
3	amedas.wni.co.jp	1360	FreeBSD	Apache/1.3.26 (Unix)
4	www.alfaoffset.se	1347	BSD/OS	Apache/1.3.26 (Unix)
5	www.sisu.ac.se	1320	BSD/OS	Apache/1.3.26 (Unix)
6	www.lobomar.se	1319	BSD/OS	Apache/1.3.26 (Unix)
7	d1o20.telia.com	1309	BSD/OS	Apache/1.3.26 Ben-SSL/1.48 (Unix) PHP/3.0.18
8	treefort.org	1298	FreeBSD	Apache/1.2.6
9	www.treefort.org	1298	FreeBSD	Apache/1.2.6
10	www.21stcenturycomputers.com	1283	BSD/OS	Apache/1.3.26 (Unix) mod_ssl/ 2.8.10 OpenSSL/0.9.6g
11	www.wycomp.com	1282	BSD/OS	Apache/1.3.26 (Unix) mod_ssl/ 2.8.10 OpenSSL/0.9.6g
12	wwwdir.telia.com	1272	BSD/OS	Apache/1.3.26 (Unix)
13	www.21net.ne.jp	1155	FreeBSD	Apache/1.3.9 (Unix)
14	www.helmarparts.com	1149	BSD/OS	Apache/1.3.23 (Unix)
15	www.lan.ne.jp	1113	FreeBSD	Apache/1.2.6
16	dbtech.net	1028	BSD/OS	Apache/1.3.27 (Unix)
17	www.icard.com.hk	1023	BSD/OS	Apache/1.3.26 (Unix)
18	www.alasearch.com	1015	BSD/OS	Apache/1.3.27 (Unix)
19	www.murrayfin.com	1000	BSD/OS	Apache
20	www.ehokenstore.com	999	BSD/OS	Oracle_Web_Listener/ 4.0.8.1.0EnterpriseEdition

* Uptime is the time since the last reboot of the front-end computer or computers hosting a site.
Source: Netcraft http://uptime.netcraft.com/up/today/top.avg.html. Accessed on 28 August 2003.

of TCO for FOSS packages have been controversial, in part because the cost structure of upgrades and maintenance is somewhat opaque relative to proprietary pricing. At acquisition, open-source solutions often cost less, depending on the type of customization and additional services that an organization chooses to buy. Deployment often requires training, which is sometimes as expensive with FOSS as with proprietary solutions, or more so. During the period of use and maintenance, where the bulk of TCO materializes, FOSS may have significant advantages. For deployment and use, costs will ultimately depend on local labour costs, which in many developing countries may be an advantage for FOSS exploitation. The availability of source code makes it possible to use in-house expertise to fix bugs or change configurations, as well as to hire external support from a competitive market that anyone can enter. What seems clear is that FOSS can help a business or public institution avoid getting locked into a vicious circle of hardware and software upgrades and changes in data formats that require investing in new license fees and significant retraining and can provoke major down time.

Ultimately, the software markets may decide which process makes better software, provided piracy, anti-competitive practices and monopolies can be curbed by government regulators. The steady growth of market share for the GNU/Linux operating system indicates that many organizations are betting that the open-source process will, over time, produce better solutions for their IT needs. Proprietary software is rarely seen taking market share away from open-source solutions where FOSS solutions exist. The final test of quality is in the numbers, and the next section describes FOSS uptake for various ICT tasks.

E. FOSS within markets

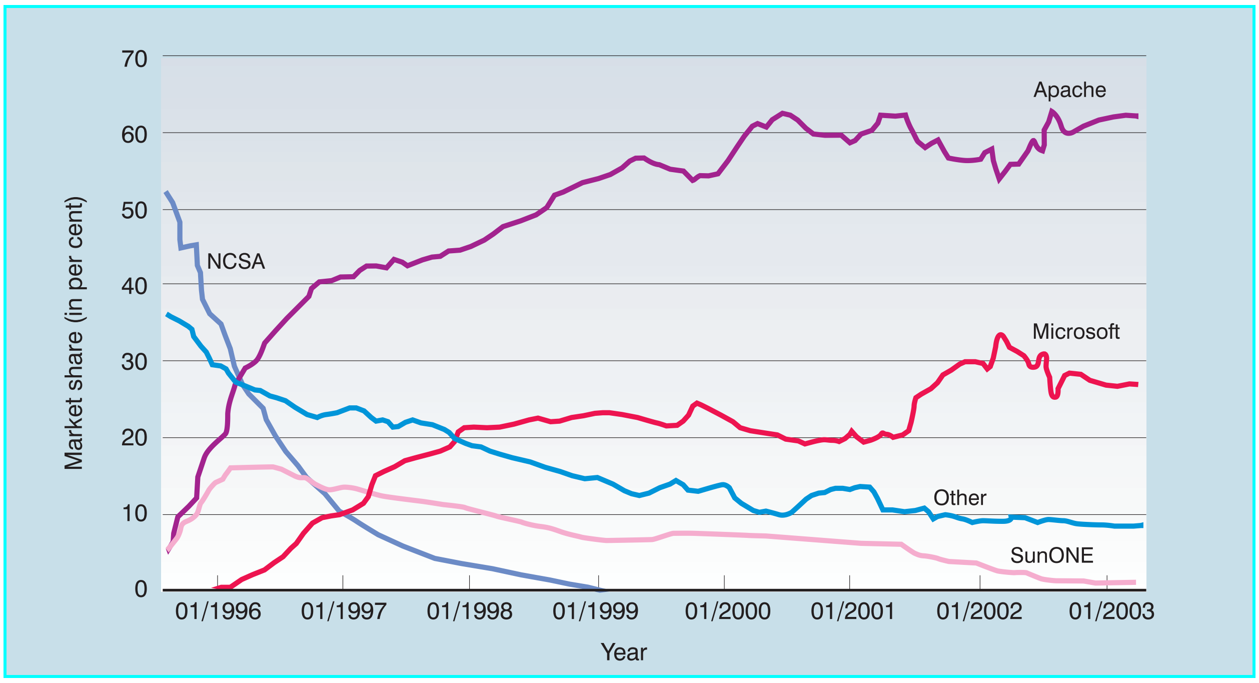
FOSS is very common, but non-expert computer users may not be very familiar with it because it has not yet made significant inroads onto the personal computer desktop in the form of an operating system or office software applications, such as word processors or spreadsheets. Typical estimates give the Microsoft Windows environment just over 90 per cent of market share, with the rest split between the Apple Macintosh and GNU/Linux-based systems. Recent IDC reports indicate that up to 15.5 per cent of businesses are considering a switch to a GNU/Linux desktop.[28] Whether this will happen is another issue. However, as many users bring their IT habits home from the workplace (along with pirated software), Linux desktop penetration in firms and government offices may generate additional growth in the household computer market. The question of desktop metrics is further complicated by the fact that, while a large number of GNU/Linux installations are downloaded from the Internet, it is not clear whether they get installed at all and, if so, where – over existing proprietary or FOSS installations, or on new computers?

Even so, many users are not aware that they may be regularly using FOSS software and data formats simply by browsing the Internet and using email, the two most common household uses of computer technology that would be unworkable without FOSS. This subsection explains why FOSS is increasingly prevalent.

The growth and, in some cases, prevalence of FOSS in important IT sectors is remarkable.[29] The open-source Web server software Apache, which sends Web pages to the computer of someone accessing a site, has dominated its market segment since 1996 and now holds at least twice the market share of its nearest competitor. A survey published in June 2003 on market share for active Web servers shows similar numbers, with Apache at 65.3 per cent.[30] Chart 4.1 shows the market shares for Web server software from 1996 to April 2003.

GNU/Linux has long been popular as an operating system[31] running computers that perform as Web servers. Recent surveys show that GNU/Linux runs 29.6 per cent of Web servers, while various versions of Windows run 49.6 per cent, with Sun's proprietary

Chart 4.1
Market share of Web server software



Source: Netcraft http://www.netcraft.com.

Box 4.3

Examples of Free and open-source software

- Open-source software is often used in mission-critical environments. Many industry standard applications are in fact open-source programmes. Following is a list of selected notable open-source programmes in addition to GNU/Linux and Apache, which are described earlier in the chapter.
- The BSD/OS/FreeBSD/NetBSD/OpenBSD[35] family of operating systems are UNIX-based, free/open-source operating systems similar to GNU/Linux. Developed at the University of California-Berkeley in the 1970s, BSD is considered one of the most secure and stable operating systems and runs a large percentage of Internet servers. The core of Apple's Macintosh operating system, Darwin, is based on FreeBSD and has remained in the open-source realm. (See table 4.2 for details of Apple's open-source activities.)
- GNU was the predecessor of GNU/Linux. It is a free version of UNIX tools created by Richard Stallman in 1984. GNU stands for "GNU is not UNIX".
- Sendmail is a free/open-source programme used for routing approximately 40 per cent of the email that travels over the Internet.
- Perl (Practical Extraction and Report Language) is a scripting language freely available for UNIX, MS/DOS, Macintosh, OS/2 and GNU/Linux, among others. Perl has powerful text-manipulation functions and is used extensively for programming Web electronic forms, and generally for generating interfaces between systems, databases and users exchanging data on the Internet.
- BIND (Berkeley Internet Name Domain) is a free/open-source programme that allows Internet domain names to be entered as text-based names instead of as IP addresses, or series of numbers, making it easier for users to reach sites on the Internet.
- The Beowulf Project is a method of connecting computers to form a high-performance computer (Beowulf cluster) that approaches "super-computer" performance. Since a Beowulf cluster can be developed from common, off-the-shelf computers utilizing FOSS, a Beowulf cluster "super-computer" can be built and implemented at a fraction of the cost of other systems with similar computing capacity.
- OpenOffice.org is a software suite that provides basic office and administrative automation. An offshoot of Sun Microsystems' StarOffice, OpenOffice runs on all major operating systems, including MS Windows, as its cross-platform functionality is based on open XML standard file formats.
- GNOME and KDE are desktop GUIs that run on top of GNU/Linux and UNIX, providing user-friendly computing to the non-programmer open-source community.
- MySOL is a relational database server.
- The Gimp is a graphics programme widely distributed with GNU/Linux. (A version for the Windows operating system also exists.) It is sometimes called "free Photoshop".

version of UNIX (Solaris) running 7.1 per cent and various BSD derivatives (which are, like GNU/Linux, open-source) running 6.1 per cent.[32]

In the last few years GNU/Linux has increasingly penetrated both the high and low ends of the enterprise market for operating systems. Nearly 40 per cent of large American companies and 65 per cent of Japanese corporations use GNU/Linux in some form, and it may now run as much as 15 per cent of the large server market overall (*Business Week* 2003). A study from October 2002 found that 59 per cent of software developers surveyed internationally expected to write applications for GNU/Linux at some point during the next year.[33] The EU-sponsored FLOSS survey (Berlecon/III 2002) found 43.7 per cent of German companies and 31.5 per cent of British companies using FOSS. It is notable that, according to several studies, Internet service providers (ISPs), large companies, small companies, and CIOs in financial services, retail and the public sector all believe that GNU/Linux use is set to increase rapidly both in their own organizations and in the market as a whole over the next several years.[34] Box 4.3 gives a more detailed overview of important FOSS available and in use today.

Amazon, E*TRADE, Reuters and Merrill Lynch are examples of multinational companies that have recently switched to GNU/Linux and Apache Web server software for their back-end computer systems. A large proportion of US Government agencies and departments, including the Department of Defense, the Department of Energy, and the National Security Agency, works with FOSS. National, state and municipal governments from China to Germany to Peru are considering, and in some cases mandating, the use of FOSS for e-government applications. A fuller description of developing countries' involvement in FOSS is presented in the survey in section H of this chapter.

E. The rationale for FOSS

While a convincing argument has been made for why someone would wish to *use* FOSS, another important question is why anyone would want to *produce* FOSS, and how this motivation translates into coherent output. Before discussing the motivations, it is worth considering some real-world indications and evidence.

1. Open evidence

Users rarely buy only software licenses; they also buy services related to the software. Organizations and firms normally buy solutions involving a combination of software, hardware and services. The services surrounding software products range from consulting, implementation, support, and training to application administration. In fact, even Microsoft has reportedly conceded that, in line with the findings of a survey by the Gartner Group, the cost of software licenses amounts to only 8 per cent of the total cost of ownership, and the other 92 per cent reflects the costs of installation, maintenance, management, and repairs after failures.[36] In what seems to be a matching estimate, Raymond (1999b) asserts that a very small portion, perhaps even less than 10 per cent, of software is developed for prepackaged retail sale. The outstanding majority consists of in-house code that is so highly integrated with firms' business and IT environments that reusing or copying the code "as is" is difficult or unfeasible.

The conclusion is clear: The majority of software development does not make money by selling licenses for prepackaged software. The opposite perception is encouraged by the fact that prepackaged proprietary software generates large revenues; however, it does so because a few producers can charge monopoly prices. For an IT services firm, the extra earnings gained by getting a commission from reselling a proprietary license may be so marginal that they may not significantly influence their choice of a proprietary over an FOSS platform for a particular client account. What should influence their choice is how precisely they can respond to clients' demands and the level of customizability, ease of maintenance and robustness a platform has to offer. From the point of view of developing countries, this issue alone is sufficient to ease worries that using FOSS platforms will diminish business opportunities.

Supporting this notion is the fact that a large part of the IT industry is developing FOSS-based activity. IBM is now a major champion of open-source software, after publicly making in 2001 a $1 billion commitment to developing technology for and reconfiguring central parts of its business models around GNU/Linux and other open-source programmes. Already in 2002, IBM announced that it had earned revenues in excess of $1 billion from sales of Linux-based software, hardware and services.[37] Other technology leaders, including Hewlett-Packard, Motorola, Dell, Oracle, Intel and Sun Microsystems, have made major commitments to FOSS for operating systems, embedded systems, cluster supercomputing, and corporate-class applications software. Table 4.3 gives a more detailed overview of mainstream IT firms' involvement in FOSS.

2. Supply motivations

Software is a digital product that can be copied an infinite number of times at zero cost, with no decrease in quality or usefulness, and is thus purely non-rival in economic terms. Freeing the source code makes software non-excludable as well, and as a result software acquires the characteristics of a public good.[38] Yet public goods normally encourage free riding. Why would people voluntarily contribute to a public good that they could otherwise use as free riders? If everyone has the same attitude, the system should unravel to the point where no one makes substantial contributions and the good never gets produced. Why do highly talented programmers choose to allocate substantial portions of their time and intellect, both of which are scarce and valuable resources, to a joint project for which they will not be directly compensated?

A great deal of effort has gone into mapping the motivations of developers. Certain studies affirm that these can be accounted for by standard economic theory. An open-source programmer's code is often precisely associated with the author and well recognized, providing a certain level of ego gratification. Personnel managers from commercial companies frequently review contributions to and participation in FOSS

Table 4.3

IT industry leaders' involvement in FOSS

Company	FOSS involvement
IBM	IBM hosts a variety of open-source projects, all under open-source licenses approved by the OSI. *http://www-124.ibm.com/developerworks/oss/*
Microsoft	Microsoft proposes a model of "shared source" as an alternative to open source. *http://www microsoft.com/licensing/sharedsource/* Microsoft Interix technology, now integrated into Windows Services for UNIX 3.0, provides an environment, under GPL license, to run both Windows and UNIX applications on a single system. *http://www.microsoft.com/windows/sfu/howtobuy/default.asp*
Pricewaterhouse-Coopers	FOSS topics are discussed on the site from a consulting perspective. *http://www.pwcglobal.com/Extweb/service.nsf/docid/30F66202E467710C85256B990072FC55*
EDS	EDS has occasional FOSS activities. Dynamator, a server page maintenance programme developed by an EDS programmer, is FOSS. *http://www.eds.com/about_eds/homepage/home_page_dynamator.shtml*
Oracle	Oracle does not have visible FOSS activities but has ported database products for Linux. *http://www.oracle.com/linux/*
Hewlett-Packard	Hewlett-Packard hosts several FOSS projects. *http://opensource.hp.com/*
Accenture	The topic of FOSS is discussed on the site from a consulting perspective. *http://www.accenture.com/xdoc/en/ideas/outlook/pov/open_source_pov_rev.pdf*
SAP	The mySAP Business Suite runs on Linux. *http://www.sap.com/solutions/netweaver/linux/* SAP DB is a free/open-source enterprise database. *http://www.sapdb.org*
Computer Associates	Computer Associates is a co-founder of Open Source Development Lab. *http://www.osdl.org*
Hitachi	Hitachi participates in FOSS projects. *http://oss.hitachi.co.jp/index-e.html*
Sun Microsystems	Sun sponsors a number of FOSS projects, including OpenOffice.org and NetBeans. *http://www.sunsource.net*
Compuware	Compuware has no FOSS activities, but the development environment shipped with its OptimalJ product is based on the open-source integrated development environment (IDE) NetBeans. *http://www.compuware.com/products/optimalj/1811_ENG_HTML.htm*
BMC Software	BMC is cooperating with The Open Group to develop an open-source Management Service Broker. *http://www.bmc.com/corporate/nr2001/032701_2.html* *http://www.opengroup.org/*
EMC	EMC has no visible FOSS activities, but development of FOSS is part of the job descriptions for currently open positions. It has also ported certain products for Linux. *http://www.emc.com/technology/auto_advice.jsp*
Cadence Design	Cadence supports open exchange among in-house developers, commercial developers and academia. Its Systems TestBuilder C++ test bench class library is available through an open-source license. *http://www.testbuilder.net* Cadence contributes to the OpenAccess coalition for standard electronic design database. *http://www.cadence.com/feature/open_access.html and http://OpenEDA.org*
Adobe	Adobe has occasional FOSS activities, mostly focusing on Python plug-ins for Adobe products. *http://opensource.adobe.com/*
Silicon Graphics SGI	SGI supports a large number of open-source projects. *http://oss.sgi.com/*
Apple	Darwin is the core of Apple's Mac OS X operating system. Based on FreeBSD, Darwin remains in the open-source domain realm under Apple Public Source License. A number of other open-source projects are supported. *http://developer.apple.com/darwin/*

projects when assessing employability. Established open-source authorities may get access to financing and attract attention from venture capital. Former open-source programmers established both Sun and Netscape. Thus, career incentives may figure prominently in motivating programmers to contribute. These phenomena, often called "signaling incentives", can appear when inputs may be judged and rewarded in one or multiple future periods even when a contract is lacking at the present (Lerner and Tirole 2000 and 2001; Holmström 1999).

Raymond (1999b) explains the open-source process as a gift economy whereby programmers make voluntary contributions as a reaction to abundance rather than scarcity, the abundance being that of knowledge and information as well as of network bandwidth and computing power. This implies the existence of win-neutral (i.e. benefit at no cost) as well as win-win (mutual benefit) and win-lose (benefit at a cost, where the cost needs to be financially reimbursed) situations.

Another way of rationalizing the existence of FOSS is the so-called cooking-pot model (Ghosh 1998). The model suggests that FOSS comes about as a direct result of the distributed structure of the Internet, where users do not want to pay or charge for goods and services that thrive on the Internet. The cooking-pot model is not a barter economy, as it does not require bilateral transactions. Further, discarding the equality between costless and valuable makes sense of the fact that the millions of people on the Internet publish on matters interesting them and contribute to communities, including those involved in FOSS software. While they will not get any cash in return, their "payment" might come in the form of complementary contributions from others, or the valuable outcomes of esteem and attention. Indeed, it has been suggested that what is increasingly scarce today is attention, while other factors, such as information and even financing, are becoming more abundant, if unevenly distributed (Goldhaber 1997).

Other studies have focused more closely on the actual FOSS developer community. The modal GNU/Linux developer appears to be a person who feels part of a technical community, who is committed to improving programming skills, facilitating his or her own work through better software, and having enjoyable and rewarding intellectual and social experiences. This person recognizes the opportunity costs of open-source programming in terms of time and money invested, but simply does not seem to value these (particularly in financial terms) as much as mainstream professionals do.[39] Individual learning, work efficiency, and collective, or "pro-social", motivations are the main reasons why these programmers choose to contribute time and effort to FOSS projects. Box 4.4 describes two recent surveys on developers' motivations.

Box 4.4

What motivates open-source developers?

A 2001 survey by the Boston Consulting Group produced additional insights by segmenting developers' responses into four characteristic groups.[40] About a third of the respondents to this survey are "believers" who say they are strongly motivated by the conviction that source code should be open. A quarter are "fun-seekers" who contribute code mainly for intellectual stimulation. About a fifth of respondents are "professionals" who work on open source because it helps them in their jobs. Another fifth are "skill enhancers" who emphasize the learning and experience they get from open-source programming. The survey also found that open-source programmers seem to cluster heavily (70.4 per cent) in the age range between 22 and 37, with about 14 per cent being younger or older. Most are not novices: more than half are professional programmers, system administrators or IT managers. (Only 20 per cent self-identify as students.)

A 2002 study sponsored by the European Union (FLOSS) surveyed about 2,800 developers online.[41] This survey reveals a group that is predominantly male and mostly under age 40. About a third of the respondents have university bachelor's degrees, another 28 per cent have master's degrees, and 9 per cent have a doctorate. The vast majority of respondents work in the IT sector for private companies or universities or are self-employed. Students make up 17 per cent of the population and unemployed developers about 4 per cent. They are widely distributed among many countries in the world, not predominantly in the United States, and exhibit high mobility as they move across national borders to work in different settings.

Each of these surveys should be approached with care, as the samples from which they gather data may be skewed by distribution of the survey instrument, inadequate response levels, and other kinds of selection bias that make accurate interpretation difficult.

3. From motivation to output

The research on individual motivation provides interesting evidence on the question of how developers think about their individual choices. But individual motivations do not by themselves add up to large-scale, coordinated action. The organization of the community has received less attention but is just as important. The vast majority of open-source projects involve a small number of developers. These projects typically depend on intensive communication and the persuasiveness of the de facto project leader to coordinate the work of the group. More explicit and formal governance structures have evolved to manage the larger projects.

What is distinctive about these governance structures is a subtle twist on decision-making authority and its relationship to hierarchy. The notion that there is no hierarchy in the division of labour lies at the heart of the open-source process. However, there can be a hierarchy of decision-making for vetting and incorporating the results of distributed work. Yet participation in that decision-making hierarchy remains voluntary for any individual developer.

The governance of the Apache FOSS project is one example. Starting with just eight people in early 1995, the Apache Group grew quickly to several dozen core developers working in loose association with hundreds of other developers who occasionally contributed ideas, code and documentation to the project. Decisions early on were made by informal email-driven consensus. This informal system came under pressure with increasing numbers of participants, and with the "burstiness" of participation: developers might be doing something else for a week before they could come back to their Apache work. However, the progress of the project as a whole could not be held up to wait for everyone's "bursts" to coincide.

The answer in practice was a system of email voting based on a minimum quorum consensus rule.[42] In 1999 the Apache Group formally incorporated as a non-profit corporation, The Apache Software Foundation.[43] It now serves as an organizational and management umbrella for a range of Web-relevant open-source projects (including the original Apache Web server as well as Jakarta, Perl, TCL and others).

GNU/Linux, as it expanded, developed a semi-formal organization for decision making about code. Clearly differentiated role structures exist within the GNU/Linux community. As the programme and the community of developers grew, Torvalds delegated responsibility for sub-systems and components to other developers, who became known as lieutenants. Some of the lieutenants onward-delegate responsibility to "area" owners whose work has a narrower focus. The organic result is what looks and functions very much like a hierarchical structure where decision making flows through a fairly well defined pyramid. The GNU/Linux pyramid works imperfectly but is evolving through trial and error towards greater scalability.

G. FOSS and development

The digital era presents significant opportunities and real risks for developing countries. One risk is being sidelined from software trends that drive the increasingly digital global economy. The combination of rapid increases in hardware processing power at declining prices and positive network externalities, whereby the value of the network increases disproportionately as it grows, suggests that markets can grow intensively and dramatically *within* the developed world, without necessarily having to expand geographically into developing countries.

As developed economies increasingly create networked purchasing and production systems that depend on advanced ICT infrastructure, countries that are not connected on favourable terms, and businesses within those countries, may be deeply disadvantaged. International organizations and non-governmental organizations are increasingly computer-enabled as well and may interact better with countries and organizations in the developing world that are similarly ICT-enabled.

This implies that decisions Governments make about procurement, standard setting and ICT adoption, technology investments, and training are critical. Over the past five years Governments around the world have begun considering legislation that would require the use of FOSS when it provides a feasible alternative to proprietary software. This phenomenon has been particularly pronounced in the developing world as these nations, struggling with limited IT budgets, look to FOSS solutions. In addition, proponents of FOSS have articulated its advantages in dealing with the mounting security concerns around networks and in providing public data accountability and transparency. Should there be any doubts as to the functionality of the data formats or processing software for critical government activities such as taxation or voting, independent experts may be requested to, without restrictions, inspect the open code and data formats. Governments have also considered the potential contribution of FOSS deployment to nascent local software industries and ICT human resource capacity building, as well as its potential spillover effects into other sectors of the economy.

Developing-country public sectors have begun to embrace the use of FOSS and encourage it in the private sector for a number of observable motivations. These motivations can be loosely grouped into three clusters: a desire for independence, the drive for security and autonomy, and new IPR enforcement. This section considers each of these motivating factors in turn.

1. Towards ICT sustainability

FOSS advocates have pointed out the technological dependency created by reliance on a few major software suppliers located in other countries. The policy debate was greatly accelerated when Peruvian Congressman Edgar Villanueva Nuñez, together with Congressman Jacques Rodrich Ackerman, tabled a bill on "Use of Free Software in Government Agencies" on 9 April 2002. Bill 1609, as it is called, would, if enacted, require all State agencies to use exclusively FOSS software in their computing systems and equipment. The Peruvian case is discussed further under "Security and Autonomy" (later in this section) and in box 4.6. A significant number of developing-country Governments have undertaken initiatives to explore FOSS. In South Africa, the Government Information Officer's Council has cited reduced costs, decreased technological dependency, promotion of universal ICT access, avoidance of proprietary software vendor lock-in and customizability to local languages and cultures as the main benefits of adopting open-source software as part of its e-government strategy.[44] In India, the Department of Information Technology, Ministry of Communication and Information Technology, is encouraging GNU/Linux and open-source software as standards in academic institutions, while the state of West Bengal is reviewing its FOSS agenda.[45] China is also examining the issue and has been providing strategic support for Red Flag Linux, a local distributor.[46] In the Brazilian state of Pernambuco, the world's first-ever law regarding the use of open-source software was passed in March 2000.[47] An extensive list of FOSS policies and initiatives is provided in the survey presented in section H of this chapter.

Countries are interested not only in the potential long-term cost savings of FOSS solutions, but also in precisely where IT expenditures are actually going. Governments should minimize their reliance on single suppliers. FOSS also helps avoid getting locked into financially disadvantageous long-term relationships with particular proprietary software vendors or producers. While the jury is still out on the cost debate, the use of free software means that installation, training, support and maintenance can be flexibly contracted out to a range of local suppliers competing on quality and price. With the use of FOSS, more domestic talent can participate in the development of local software. This makes it possible to keep IT expenditures, as well as experts and promising young talent, at home and contributing to a nascent local software industry. At the same time there is a motivation to upgrade the country's human resource capacity and technological skill base.

FOSS eliminates the national-level economic loss resulting from duplication of work, in particular if such development has been done in a public or academic institution. Sharing applications and their source code across ministries, government offices and schools and universities can be a public policy stance. A variety of positive spillover effects to other technology and non-technology sectors are also possible and are discussed in box 4.5.

Finally, promoting FOSS can have an anti-monopolistic effect on the IT market and industry in a country. Network externalities in the software industry, whereby the value of a product such as a word processor or operating system increases with the number of people using it, may result in monopolies with inferior products. The prevalence of a particular software application is seen as a dominant quality in itself, and this can motivate developers to port new programmes or upgrades specifically to it, regardless of its underlying technical qualities. FOSS allows anyone to provide IT services and thus reduces barriers to entry. While a certain open-source programme may come to dominate its market niche, no particular institution or business can use it to build a monopoly market position.

Hesitation, in particular among accustomed users, should be expected if a Government decides to move away from existing proprietary solutions. However, ease of use, bred through familiarity, may seem less

Box 4.5

Open-source processes outside the software sector

Two notable areas where open- and free-source philosophies are making inroads are publishing and biology, in particular in the research work on the human genome.

Open-source publishing is often referred to as open-content publishing. Open content is the content production process together with the content itself, when it is distributed according to an open-content license agreement. The basis for open-content licensing is that content is freely available for modification, use and redistribution, with certain restrictions aimed at supporting its freedom from the threat of proprietary closing (Keats 2003). A number of open-content directories and projects have sprung up,[48] inspired in part by dissatisfaction among teachers and lecturers with the rising cost and decreasing quality of new editions of textbooks.[49] In the development context, given the cost of content as well as the under-funding of schools and lack of expertise in many countries, collaborative development of content in an open environment and process could improve access to high-quality, locally relevant content. Open content has great potential to contribute to a "knowledge commons" that can positively affect economic development. Governments and the UN system could contribute to a reusable global body of knowledge by declaring many of their publications, documents and other content, produced with members' contributions, government or public funds, to be open content.

The FOSS programme that allowed the public Human Genome Project at the Sanger Institute to assemble the genome, in parallel with Celera's commercial effort, ensured that the human genome data would remain in the public domain.[50] Jim Kent wrote the programme to stop the genome data from getting locked up by commercial patents. This situation demonstrated the need to think about more than just open-source code; in the scientific community there is awareness of the importance of open data and procedures, as replicability is the only guarantor of scientific validity.[51] However, there have been assertions that without a public open-source competitor, the human genome may still be in the proprietary domain, available to those capable of paying for a subscription to what many consider the common knowledge of humanity.

Other organizations have been mimicking the FOSS model as well. Bioinformatics.org affirms in its mission statement that it aims to "promote freedom and openness in the field of bioinformatics [and] hopes to lower the barrier to entering and participating in the field of bioinformatics, as access to cutting-edge resources can be prohibitively expensive for those working individually, in small groups, at poorly funded institutions or in developing nations".[52] In another example, the Alliance for Cellular Signaling will build a virtual cell that will allow scientists to perform experiments completely on their computers. Replicating the FOSS process, several laboratories will act as central coordinators, and hundreds of researchers are expected to contribute over the Internet..[53]

advantageous when new licenses have to be purchased for upgrades that in turn often require corresponding upgrades in hardware.

2. Security and autonomy

Security of public data is a leading concern of Governments, particularly in the wake of recent worldwide computer virus attacks and growing fears of cyberterrorism and cybercrime, as well as spyware.[54] At a minimum, introducing diversity into the base of functioning software code reduces the possibility of catastrophic failures caused by viruses that attack a software monoculture. Finally, because Governments cannot choose their customers or citizens, it follows that they should not oblige them to use costly proprietary software and closed data formats.

The need for open public data formats is directly relevant to calls for increased accountability and transparency in public sector governance. As was mentioned earlier, Peruvian Congressman Edgar Villanueva introduced a bill to mandate the use of FOSS in public administration. In an exchange of letters with Microsoft Peru,[55] he stressed that, to guarantee free access by citizens to public information, it is indispensable that the encoding and processing of data not be tied to any single provider. The use of standard and open formats guarantees free access. If one is to guarantee the permanence of public data, the usability and maintenance of software should not depend on the goodwill of suppliers or on conditions imposed by them in a monopoly market. At a fundamental level, nations must, in order to guarantee national security, be able to rely on systems without elements controlled at a distance. Box 4.6 provides a summary of the

Box 4.6

Summary of main points of E. Villanueva's letter to Microsoft Peru

Bill Number 1609 (The Use of Free Software in Public Administration), introduced by Congressman Edgar Villanueva, is intended to require the use of FOSS in all government systems, when there is a choice between FOSS and proprietary software.

Congressman Villanueva's letter to Microsoft Peru (8 April 2002) expressed the following principles:

- To guarantee free access by citizens to public information, it is indispensable that the encoding of data not be tied to a single provider. The use of standard and open formats guarantees free access.
- To guarantee the permanence of public data, the usability and maintenance of the software should not depend on the goodwill of suppliers or on monopoly conditions imposed by them.
- To guarantee national security, the State must be able to rely on systems without elements controlled from a distance. Systems with open-source code allow the State and citizens to inspect the code themselves and check for back doors and spyware.

In response to the concerns raised by Microsoft Peru, Congressman Villanueva argues the following:

- The bill does not meddle in private-sector transactions and protects equality under the law (i.e. nobody is denied the right to offer these goods to the State). There is no discrimination, since the bill specifies only how the goods are to be provided, not who has to provide them. Proprietary software companies are free to offer FOSS solutions to the Government in a competitive tender.
- The bill stimulates competition, since it tends to generate a supply of software with better conditions of usability, and to enhance existing work, in a process of continuous improvement.
- Proprietary software creates mainly "technical tasks of little aggregate value" in countries like Peru; free and open software creates more technically qualified employment, stimulates the market, and increases the shared fund of knowledge, opening up service alternatives to the benefit of producers, service organizations and consumers.
- As for security, bugs in free software are rarer and are fixed much more quickly than in proprietary software.
- Free software in no way implies ignorance of intellectual property laws; the great majority of free software is covered by copyright.
- The bill is not mistaken regarding the costs of free software: while the possibility for savings in payments for proprietary software licenses is mentioned, the foundations of the bill clearly refer to the fundamental guarantees to be preserved (free access, permanence and security) and the stimulus to local technological development.
- The use of free software contributes significantly to reducing life-cycle costs: support and maintenance can be freely contracted out to a range of suppliers competing on quality and cost for installation, enabling, support and maintenance; maintenance carried out is easily replicable without incurring large costs, since modifications can be included in the common fund of knowledge; and the huge costs created by non-functioning software are reduced by using more stable software, which is one of the virtues of free software.
- Migration to new systems is in fact cheaper when FOSS is used, since all data are stored in an open format.
- Interoperability is guaranteed as much by standard formats (as required by the bill) as by the possibility of creating interoperable software given the availability of the source code.

positions taken in response to Microsoft's discussion of the disadvantages of legislating against proprietary software use in Peru's public institutions.

The need to have public and open standards for software applications and data files that handle public information is now universally accepted. Software that is used to handle public records, taxation or, in the future, voting may need to follow FOSS standards. Further, Governments need to maintain certain key public data and be accountable for its processing. With closed-source proprietary software and data file formats, should the vendor choose to discontinue support for technical reasons (e.g. because maintaining backward compatibility is burdening the source code of current and new versions) or financial reasons (e.g. an unsatisfactory revenue stream or bankruptcy), public offices may find themselves forced to upgrade hardware or software (often both) or convert to another system, with the resulting cost implications.

A study on government use of FOSS in Europe (Berlecon/III 2002) expresses many of the same concerns as Congressman Villanueva. It argues that FOSS, by its nature, better fulfils government responsibilities such as satisfying the public's right to access certain information and know how that information is processed, and maintaining the security and permanence of public data.

Other developing countries have also expressed dissatisfaction with the proprietary software development and marketing model, in particular pointing to the negligible influence they, as "smaller" customers, have on how software develops. FOSS is expected to provide more flexibility and allow more autonomous input into software development. This can be conceived of as an ownership issue: developing nations desire an opportunity to articulate their software needs and participate in the innovation process as end users of software products. In addition, they welcome the possibility that an indigenous industry can participate in both identifying and meeting those software development needs.

3. Intellectual property rights

With increased emphasis on and pursuit of IPR enforcement at the international level, the choices available to software users are becoming more distinct. As countries move away from the gray options of software piracy toward a stricter implementation of standard intellectual property rules, this forces real choices. While proprietary desktop software is still largely considered more user-friendly than the alternatives, its market penetration and price are not related in countries where software piracy is commonplace. Thus, all efforts by international proprietary software producers to decrease piracy in fact improve the fundamental conditions for increased adoption of open-source software.

It should also be kept in mind that, historically, the basic precondition for the appearance, as a concept, of IPR and law with regard to creative goods and services has been the high cost of reproducing the carrier mediums (printed books, vinyl records, film stock and magnetic and optical digital media), not the ability of states and Governments to enforce the accompanying legislation. Technology has relegated this condition to history's dustbin, and Governments are now faced with acting on their own law, something that was not a practical consideration until a few years ago.

Conversely, to think FOSS presents an alternative to respecting IPR is a gross misunderstanding. In fact, FOSS requests users to, without exception, respect the intellectual property of the software's author(s) as outlined in the enclosed GPL or OSD license, and it needs Governments to provide legal protection and remedy when this is necessary and deserved. The full text of the GNU General Public License and the criteria for OSD licenses are contained in annexes I and II of this chapter.

There is a broader issue here for Governments than simple tolerance (or lack thereof) of a certain degree of software piracy. The question is what regime for ownership and distribution of IT tools best serves the interests of developing countries and of the global economy as a whole. To think of FOSS as simply a less expensive alternative to proprietary software misses an important aspect of what FOSS enables. In an FOSS environment, the degree to which a software tool can be utilized and expanded is limited only by the knowledge, learning and innovative energy of the potential users and not by exclusionary property rights, prices or the power of countries and corporations.

The current debate often pits proprietary licensing against the GPL. Commercial software producers argue that promoting the GPL means locking out any software development from possible future commercialization. As the previous section indicated, the bulk of software revenues come from customization, servicing or hardware, or all of the above bundled in solutions. Indeed, IBM *did* earn $1 billion on the back of GPL GNU/Linux. Finally, proprietary licensing allows only the owner to commercialize the intellectual property at stake and makes it inaccessible to anyone else. Anyone seeking to redistribute a derivative version of a proprietary programme would be prohibited from doing so under the terms of the license. Thus, the formal outcome is not that different from that of the GPL (Lessig 2002). In terms of ICT strategy and its relation to innovation and development, there have been indications that the proprietary model may encourage excessive copyrighting and patent hoarding, with the final outcome being reduced investment in research and development (R&D) activities and a decline in innovation as funds for R&D are redirected towards patent acquisition and royalty payments (Bessen 2002, Bessen and Hunt 2003).

FOSS presents a significant development opportunity because of the critical role that users can play in determining new products and the overall trajectory of technology evolution. Software innovations can and should come increasingly from developing countries.

Developing countries are not implicitly stuck relying on commoditized, hand-me-down innovation from the developed world. In an FOSS environment, their own lead users could push technology development towards applications that specifically fit local needs and demands. However, for indigenous demand to be expressed, users need to understand the possibilities they have and the ways in which a digital infrastructure could contribute to their lives.

When those possibilities are evolving as quickly as they are today, it seems certain that IT consumers generate demand primarily through a process of learning by doing.[56] With increasing familiarity, users can gradually come to understand what the technologies can do for them, and then to imagine new possibilities, provided they are fully aware of their options and the existing technical options. Thus, an environment where software is normally used under restrictive intellectual property licensing may not be the most conducive for ICT development and bridging the digital divide. The empowerment that comes with free access to source code is not a simple price advantage, but may rather be a necessary economic prerequisite for evolving demand. The applications that find widespread acceptance and drive technology and infrastructure deployment forward in developing economies may certainly come from within those countries.

H. Policy options for FOSS

There are two general areas of policy implementation options to be considered by Governments, each with different public-sector, civil-society and private-sector dynamics. Each of these potential paths has constraints or obstacles that developing countries in particular must be aware of when considering the various policy options available to them in adopting FOSS.

- **Formal vs. informal approaches:** Formal approaches such as legislation or a government strategic plan may be weighed against more informal, flexible approaches to letting FOSS use evolve without normative patronage.

- **Strategy and level of involvement:** Strategy initiatives may be carried out at sub-national, national or regional levels, and they may also entail different degrees of involvement, from awareness building to procurement to funding of R&D.

These options are not mutually exclusive but rather represent spectrums along which Governments can choose to array specific policies or a more general approach to FOSS use. The relationship between government, civil society and industry may also be varied, with initiatives coming in a mixture of strengths from any given stakeholder. There is no prescription or tried-and-tested scheme: policy makers will have to consider their national circumstances and ICT development priorities. This chapter considers several options and offers examples of applications throughout the world.

1. Formal involment

A number of Governments have pursued formal approaches to the adoption of FOSS in the public sector, considering legislation to mandate the use of open-source solutions in government applications or at least seriously consider them as an alternative to proprietary software. In the industrialized world, this trend has been strongest in Europe, particularly France and Germany. The French Parliament proposed a bill concerned with both the use of open standards and the availability of source code for software used by the Government. An Italian bill under consideration mandates a preference for FOSS in all government offices, and a Spanish bill requires regional governments to prefer and promote open-source products. In April 2002, the administration of the Spanish district of Extremadura put in place a plan to switch all computer systems in government offices, businesses and homes to Linux and FOSS applications.[57] The Government of the United Kingdom has set out policy to consider open-source solutions alongside proprietary ones in IT procurement; to use products that support open standards and specifications in all future IT development; to consider obtaining full rights to bespoke and customized software code for proprietary software it procures; and to explore further the possibilities of using FOSS as the default exploitation route for government-funded R&D software.[58]

A number of Latin American governments at the national and local levels have introduced or passed and introduced legislation on the use of FOSS solutions in the public sector. The Peruvian case was discussed above. Argentina's Parliament reviewed a proposal that mandates, with a few exceptions, the use of FOSS in all government offices and state-owned enterprises, but the Parliament collapsed in the fiscal crisis of 2001 before a decision was taken on this bill. In Brazil, four cities – Amparo, Recife, Ribeirao Pires and Solonopole – have passed laws giving preference to or requiring the use of FOSS, and other municipalities and states, as well as the national Government, have considered similar legislation.

Other countries have taken slightly less formal steps towards using FOSS in government. France, in addition to considering legislation, has created an Agency for Technologies of Information and Communication in Administration (ATICA), which seeks, among other things, to encourage the use of free software and open standards.

A less formal, more flexible approach has its advantages. Foremost of these is the value of allowing the FOSS phenomenon to develop by itself, along with the attendant organizational innovation that could bring. Different user communities have the opportunity, through the open-source process, to come up with unique and contextually appropriate technological and organizational models, building indigenous commitment and ownership along the way.

It is often argued that Governments do not have an enviable record of successfully legislating and promoting industrial policy and that thriving software developments are best left alone (Evans 2002). While this issue may have some relevance in countries with developed market economies, within the development context one cannot help but wonder whether the market purist and non-interventionist concepts of the Washington consensus are for export only. A different issue is, if a Government does decide to adopt a pro-FOSS legislative bias, how this should be implemented and how formal it should be from a normative viewpoint.

2. Strategy and direct involvement

Since Governments are important consumers of ICT in developing countries, their participation is crucial for the success of any open-source initiative. Government can be involved at the level of strategic policy, building awareness and promoting conscious and informed choice among its administration as well as industries and civil society. It may act as a procurer, and it may directly finance R&D. This section considers different levels at which Governments can implement an FOSS strategy.

A good example of high-level strategic thinking is the case of the Government of South Africa. A council to consider the use of FOSS was convened in early 2003. The council delivered an official recommendation promoting the use of open-source applications when proprietary alternatives did not offer a compelling advantage. The recommendations were formulated at a strategic level and are described in box 4.7. The advantage of a strategic approach lies in the nature of

Box 4.7

Summary of strategic steps highlighted by South Africa's government council on open-source policy

South Africa's Government Information Technology Officer's council's FOSS strategy includes steps to consolidate and expand the capacity needed to implement and support FOSS solutions, including:

- Provision of information to key decision makers (bearing in mind the need to demonstrate convincingly the security measures and business principles of FOSS)
- Generation of expert advice on the suitability of FOSS solutions
- Trouble-shooting for newly implemented FOSS solutions
- Software development assistance
- Training for FOSS developers and users (concentrated in existing learning institutions)
- Development of a research programme to enable optimal understanding of and decision making regarding FOSS (built on the networking nature of the FOSS development model)
- Creation of FOSS support structures (some institutional development will be necessary)

Source: Open Source Software in Government, www.oss.gov.za.

software provision. As a complex knowledge product, software requires a technological and social infrastructure to facilitate its provision. A strategic approach would allow Governments to work in collaboration with donors to map out potential areas for development assistance, in particular identifying potential human resource capacity-building and technical assistance needs.

The report recommends creating strong linkages with higher education institutions to build a national collaborative network that can be extended internationally. It also emphasizes building partnerships within the public and private sectors and civil society, as well as regionally within Africa and globally. The strategy emphasizes the importance of building support among key stakeholders, including the political level, senior management, IT professionals and government users.

Still at the strategic level but with the stakes raised to international collaboration, FOSS may have the potential to generate large economies of scale and positive spillover effects in regional capacity building and infrastructure development. A number of regions have taken steps toward collaborating on FOSS, and such cooperation has been most pronounced in Africa. In early 2003, African countries from across the continent launched the Free and Open Source Software Foundation for Africa (FOSSFA), an organization aimed at promoting the use of FOSS throughout the continent.[59] Box 4.8 presents FOSSFA's recommendations for formulating a policy regarding FOSS use.

FOSSFA anticipates that FOSS will provide opportunities to develop local programmes built by Africans for use in Africa. Regional organizations such as FOSSFA thus see the development value of open source in broad terms. An important aspect of such strategies is to emphasize the capacity-building dimension associated with open-source technology. Regional organizations have the potential to work with educators on a broad scale to introduce open source into schools where young people can learn to use, maintain and modify software. The vision for the future is one of a regional technical revolution of sorts, in which Governments and the private sector embrace FOSS and can rely on regionally developed software and expertise.

As far as practical measures are concerned, a number of Governments have piloted the deployment of FOSS in government service delivery agencies at the subnational level. In South Africa, for example, some provinces and national departments are using GNU/Linux and other FOSS applications on a trial basis, and the Department of Health has implemented an FOSS health information system in both national and provincial departments that is now also used in some other African countries.

Some European Governments have begun shifting serious national-level support to open source. For example, France's ministries of Defense, Culture, and Economy have shifted to open-source operating systems. Germany's Federal Institute for Agriculture and Food has installed open-source operating systems on servers and workstations. In Britain, the National Health Service has adopted an open-source standard.[60]

Some developing countries have seen the private sector taking the initiative in cooperating with the Government in open-source software development. In India, for example, while government agencies have begun to explore the potential of FOSS applications, especially in education, private entrepreneurs have developed the Simputer, an FOSS-based handheld device. (See chapter 3 for a discussion of the Simputer.) Collaboration between the public and private sectors is essential to a successful systematic adoption of FOSS solutions. The Simputer demonstrates that innovative private-sector FOSS solutions are possible. Yet even in this case, the developers anticipated needing government assistance to help disseminate the device. They realized that the Government would have to act as a major consumer in order to achieve the necessary critical mass for popularizing the product.

Some countries have more explicitly encouraged collaboration between the public and private sectors in the production and adoption of open-source applications. Attempting to encourage the continued development of the local software industry, the Government of Germany has struck a deal with IBM that offers government offices discounts on IBM computers with preinstalled GNU/Linux software provided by the German GNU/Linux distributor SuSE. Singapore, through its Economic Development Board, which is charged with executing strategies to boost the Singaporean economy, is offering tax breaks to companies that use the GNU/Linux operating system instead of proprietary alternatives.

Finally, one needs to look at the issue of direct funding of FOSS project and development needs. A number of examples are listed in the survey that follows. An important question is whether, for software

Box 4.8

The Free and Open Source Software Foundation for Africa policy recommendations for FOSS

In its Action Plan for June 2003 – June 2005, FOSSFA proposes three distinct approaches a Government might take in formulating its FOSS policy. It argues that any particular country should seek the mix of these approaches that best reflects its ICT and development reality. (See annex III of this chapter for the text of FOSSFA's founding statement.)

1. ***The neutral approach***

- Governments can adopt a neutral approach ensuring that choice is supported and discrimination against FOSS is eliminated. Governments should:
- Adopt policies to ensure that FOSS is carefully considered in IT procurement processes.
- Implement criteria for evaluating open-source products, and procedures for adopting and maintaining open standards.
- Allow open-source software to compete on an equal basis with proprietary alternatives.
- Initiate communication to enhance knowledge and understanding of FOSS.

2. ***The enabling approach***

- In an enabling approach, policies are geared towards creation of the capacity to use FOSS. Governments should, in addition to the neutral approach:
- Develop the capability to give guidance on selecting and implementing FOSS.
- Promote education and training for the use of FOSS products.
- Support the establishment of partnerships between users and developers.

3. ***The aggressive approach***

In an aggressive approach, Governments actively encourage the development of FOSS through both legislation and policy. Governments should:

- Actively support FOSS developers' communities and projects.
- Adopt strategies to increase commitment to open-source products.
- Conduct regular auditing of the impact of FOSS on government service delivery.
- Participate in programmes that can minimize risks associated with FOSS.
- Standardize FOSS where analysis shows it to be the best alternative.

Source: FOSSFA Action Plan 2003-2005, www.fossfa.org/resources.html.

produced with public funds, there should be any preference for a specific licensing model. Policy makers should scrutinize the available OSD licenses as well as the GPL and reflect on the details of the debate between Microsoft Peru and Peruvian Congressman Villanueva. While there is sometimes a temptation to prefer the "copyleft" spirit of the GPL, let it be noted that the very successful Apache server software and the BSD operating system are distributed under less restrictive OSD licenses that actually allow proprietary use of the source code. Yet these programmes remain the frontrunners in their domains.

3. Examples of FOSS policy action in developing countries

The following are examples of FOSS use in developing countries. Where relevant, the policy framework is described and the main forms of involvement are noted. The survey is not comprehensive and is based on information found by conducting keyword searches on the Internet.

Argentina[61]

- A bill "Policy for Free Software Use for the Federal State" presented to Argentina's House of Congress in April 2001 called for mandatory government use of FOSS. The economic crisis forced the Government out before a vote could be taken. A similar bill was submitted in March 2002 and is under review.
- The current bill proposes FOSS as a component of the national campaign against software piracy.

Brazil[62]

- Rio Grande do Sul was the first administration to pass a law making FOSS use mandatory in both government agencies and non-government-managed utilities.
- Four cities in Brazil have passed legislation requiring preference for "software libre" where an open-source option is available.
- The national health care system plans to release 10 million lines of source code.
- The first annual Free Software International Forum was held in Brazil in May 2000.
- In the province of Pernambuco, the world's first law regarding the use of open-source software was passed in March 2000.

China[63]

- The Government-supported China Academy of Science together with Government-owned Shanghai New Margin Venture Capital established Red Flag Linux, a Chinese-language Linux distribution.
- The Beijing Software Industry Productivity Center was established by the Beijing municipal government and has launched a project named "Yangfan" to improve the performance of local distributions of GNU/Linux.
- The strong presence of international FOSS developers, including Turbo Linux, Red Hat and IBM, is noticeable.

India[64]

- A growing attraction to Linux in India has persuaded Microsoft to share source code with a particular government body.
- The Simputer was developed by a group of scientists from the Indian Institute of Science and Encore Software. (See box 3.3 in chapter 3.)
- Government agencies promote the use of localized solutions such as Indian-language computing. The Centre for Development of Advanced Computing and the Department of Information Technology are supporting the development of a Hindi GNU/Linux distribution called Indix.
- The Department of Information Technology has expressed an intention to introduce Linux as the de facto standard in academic institutions; research establishments will develop distributable toolboxes; central and state governments will be asked to use Linux-based offerings.
- The West Bengal Electronics Industry Development Corp Ltd., the state's nodal IT body, has formed a Linux cell to support various government IT projects inside and outside the state.
- Talks with major FOSS industry players on joint projects are in progress.

Malaysia[65]

- The Government committed in November 2001 to using FOSS in key agencies, such as the Treasury, and in areas such as e-procurement.
- The Malaysian National Computer Confederation operates an FOSS special interest group.
- The Prime Minister launched the Komnas (Komputer Nasional) Twenty20 Personal Computer, built on FOSS by the private sector.
- The Malaysia Institute of Electronic Systems, the ICT advisor to the Government, is pushing the shift towards FOSS, including an attempt to build a low-cost PC based on GNU/Linux.

Pakistan[66]

- The Government Technology Resources Mobilization Unit has created a "Linux Force" task force that is expected to help Pakistan move toward FOSS. This would include funding for R&D programmes for client software, training and local-language application development.

Peru[67]

- Congressman Edgar Villanueva has introduced Bill 1609, "The Use of Free Software in Public Administration", to mandate the use of FOSS in all government systems.
- Congressman Villanueva's open confrontation with Microsoft Peru has earned him and Peru the reputation of being the developing world's FOSS radical.

The Philippines[68]

- Bayanihan Linux, developed under the Open Source Project of the Advanced Science and Technology Institute of the Philippines, has had its second release and is bundled with the latest office suite, image and text editors, Internet and networking tools and multimedia applications. Bayanihan is a single-CD installation tailored to local demand.

Republic of Korea[69]

- The local company HancomLinux signed a deal in January 2003 with the country's Central Procurement Office to supply the Government with 120,000 copies of its Linux desktop office productivity software, HancomOffice. The open-source software, which is compatible with Microsoft's Office applications, including Word and Excel, is expected to save the Government money in the long run and stimulate business for local companies competing against Microsoft in the software industry.

South Africa[70]

- A Government council convened to consider the use of FOSS published an official recommendation promoting the use of open-source applications when proprietary alternatives do not offer a compelling advantage, and highlighted the necessary strategic steps.
- In January 2003, the Government declared that it would use FOSS and set up a council for scientific and industrial research to help develop programming skills.
- South Africa has taken the lead in regional collaboration on OSS, including the Free and Open Source Software Foundation for Africa.

Thailand[71]

- The Government-supported technology development group NECTEC has developed a GNU/Linux distribution for schools and government desktops and servers – the Linux-SIS (School Internet Server) for servers and the Linux TLE (Thai Linux Extension) for government desktops. The project aims to narrow the gap between use of pirated and legal software, and to promote local business development.

Viet Nam[72]

- Government delegates to a software seminar in Hanoi concluded that Viet Nam could save hundreds of millions of dollars annually and better guarantee information security by switching to FOSS.
- Vietnamese IT companies are working on FOSS projects by subcontracting with foreign companies
- FOSS was included in the National Program on Information Technology.

I. Conclusions

The Internet, or the rapid introduction into human affairs of extensive telecommunications bandwidth configured as a neutral and public network, changes some very important things about the constraints and opportunities that individuals, organizations and countries encounter as they move towards increasingly knowledge-intensive economies. Developing countries will simultaneously confront new and old problems: the promise of information-enabled development; the challenge of managing complex, technologically embedded relationships with multinational firms and the developed world; and the question of how to configure IPR regimes that are increasingly crucial pillars of economic growth. The advantages for developing countries of promoting policy that will provide a positive environment for open-source IT are manifold, and any differences in comparison with the developed world are generally ones of degree, not of direction.

FOSS is here to stay for the foreseeable future. Experience so far has shown that open-source environments often produce reliable, secure and upgradable software at a relatively low cost. By definition, FOSS provides an improved approach to security issues and to the need for public and open standards, a subject of concern in government institutions. Open source eliminates the national-level economic loss that otherwise results from duplication of software development, in particular if it has been done in a public or academic institution. Supporting FOSS can have an anti-monopolistic effect on the IT market and industry in a country and globally, thereby reducing the threat of technological and financial lock-in.

Governments, after considering the experience of those developing and developed countries that have initiated FOSS policy and activities, should decide which approach suits their environment best. While some countries may have large numbers of technically qualified and interested experts, this may not necessarily be the case throughout the developing world. Thus, government policy on human resources for ICT development may need to include an FOSS agenda. While its low cost does not drive the development of FOSS globally, in developing countries it may well speed adoption, particularly given the increasingly stringent enforcement of IPR demanded by proprietary software producers. Money spent on licenses may be better used in training ICT experts who can perform real software development, rather than just "click on the menu". Finally, the increasing adoption of FOSS in the developed world is creating export opportunities for customized software from nascent IT industries in developing countries.

Ultimately there are many different ways to manage the transition to a knowledge or information economy. But if the production, flow and control of information are defining features of a community, an economy and a society, then the rules that govern information become foundational. Software is one of the most important sources of those rules. As with any set of rules, what matters is not just what the rules say but how they come to be written and who can change them under what conditions. FOSS should be seen, then, as more than just a different kind of product. It is a different kind of process for building, maintaining and changing the rules that govern information flows.

Notes

1. Moving a software programme from the operating system environment in which it was developed to another operating system environment; writing a version that runs on another system.
2. For a critical evaluation of the data see Tuomi (2002).
3. The competition as indicated by Hoover's (www.hoovers.com) consists of the following 10 firms in order of revenue size: Siebel, BMC Software, Novell, Network Associates, Activision, Sage Group, Infosys (Bangalore), Business Objects, Legato Systems and RSA Security.
4. The term *packaged* means that the software is written for mass retailing and is not customized for specific user needs. It typically includes operating systems, utilities, applications and programming languages.
5. See *www.businessweek.com/magazine/content/03_02/b3815723.htm.*
6. Total revenues for IBM, Sun Microsystems, EMC, Dell, Hewlett-Packard, Gateway, Apple, Fujitsu and NEC approach $276 billion, but it is difficult to determine what portion comes from software-related activities.
7. This analogy has been attributed to Mitchell Stoltz of Mozilla.org.
8. It may be included in the software package or on the CD-ROMs, or it may be posted on the Internet and its location (URL) indicated.
9. A frequently asked question is why software developers would choose to share source code in this unprotected, non-proprietary fashion. This discussion has a number of economic and sociological aspects and is discussed in some detail in section F of this chapter.
10. The implied meaning of "Fordist" has its origins in Adam Smith's discussions on the division of labour. The manufacturing process for any product can be broken down into steps, and having each worker specialize in one of these steps leads to substantial productivity gains. The approach was perfected in Ford's automotive factories, thus the term.
11. One of the most pervasive and detrimental legacies of the Internet hype of the late 1990s was the popularity of an over-simplified idea of "self-organization". New technologies seemed to be undermining or at least presenting alternatives to traditional command-and-control hierarchies in business, government and elsewhere.
12. Ken Thompson is usually given credit for being the "inventor" of UNIX and Dennis Ritchie is given credit for C. Both were employees of Bell Labs.
13. The group at the University of California at Berkeley was particularly influential. Bill Joy, who would go on to found Sun Microsystems, headed the first Berkeley Software Distribution (BSD) project of UNIX in 1978.
14. This case would drag on for 13 years before finally being dismissed by the Reagan administration in 1981. See DeLamarter (1986).
15. In Stallman's view, "the sharing of recipes is as old as cooking", but proprietary software meant "that the first step in using a computer was a promise not to help your neighbor". He saw this as "dividing the public and keeping users helpless" (1999, p. 54). For a fuller statement see *www.gnu.org/philosophy/why-free.html.*
16. For a description of GNU see box 4.1 in the text.
17. GNU.org, at *www.gnu.org/licenses/gpl.html.*
18. There have been several modifications to these specific provisions, but the general principle is unchanged.
19. In the Acknowledgements section of the *Open Sources: Voices from the Open Source Revolution* (1999) omnibus, Emacs is described at some length: "Calling Emacs editor an editor is like calling the Earth a nice hunk of dirt. Emacs is an editor, a web browser, a news reader, a mail reader, a personal information manager, a typesetting program, a programming editor, a hex editor, a word processor, and a number of video games. Many programmers use a kitchen sink as an icon for their copy of Emacs. There are many programmers who enter Emacs and don't have to do anything else on the computer. Emacs, you'll find, isn't just a program, but a religion, and RMS (Richard M. Stallman) is its saint."
20. See *http://gcc.gnu.org* for more details.
21. See *http://sources.redhat.com/gdb/* for more details.

22. See *http://opensource.org/osd.html* for more details.

23. "We think the economic self-interest arguments for Open Source are strong enough that nobody needs to go on any moral crusades about it". See *www.opensource.org* for more details.

24. Gomes L (1998), Microsoft acknowledges growing threat of free software for popular functions, *Wall Street Journal*, 3 November: B6; and the "Halloween Memo", at GNU/Linux.miningco.com/library/blhalloween.html.

25. An unauthorized text of the so-called Halloween Memo can be found in unabridged format at *www.scripting.com/misc/halloweenMemo.html.* The OSI has posted the leaked version of the memo with commentary at *www.opensource.org/halloween/halloween1.php.* The OSI reported that the Microsoft Halloween Memo explicitly stated:

    "OSS is long term credible…. [because] the real key to GNU/Linux isn't the static version of the product but the process around it. This process lends credibility and an air of future-safeness to customer GNU/Linux investments. GNU/Linux has been deployed in mission critical, commercial environments with an excellent pool of public testimonials…. Recent case studies provide very dramatic evidence that commercial quality can be achieved/exceeded by OSS projects. The Internet provides an ideal, high visibility showcase for the OSS world. The ability of the OSS process to collect and harness the collective IQ of thousands of individuals across the Internet is simply amazing. More importantly, OSS evangelization scales with the size of the Internet much faster than [Microsoft's] evangelization efforts appear to scale."

    In 2002 and 2003 Microsoft began experimenting with allowing limited viewing of its source code to large customers and Governments that in particular may wish to audit for security concerns, under particular agreements relating to non-disclosure and non-competition.

26. See note 3.

27. For a discussion of an empirical test of bugginess that compares FOSS and a proprietary platform, see Kuan (2003).

28. See *www.computerworld.com.au/index.php?id=2110919358&fp=16&fpid=0* .

29. Market share numbers for software should always be read cautiously, as sampling and measurement issues complicate any straightforward assessment of who is actually using what software in these highly distributed markets. The data discussed here come primarily from industrialized counties. Market share data for developing countries are not currently available.

30. E-soft, *www.securityspace.com/s_survey/data/200303/index.html.*

31. Operating systems perform basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers. Most computer users are familiar only with the Microsoft Windows operating system.

32. See *www.netcraft.com/Survey/index-200106.html#computers*; see also *www.oss-institute.org/reference.html.*

33. See *www.businesswire.com/cgi-bin/f_headline.cgi?bw.111301/213170209.*

34. See *www.dwheeler.com/oss_fs_why.html.*

35. Berkeley Software Distribution.

36. This concession comes from the letter that Microsoft addressed to Peruvian Congressman Edgar Villanueva, arguing against his ambition to legally designate FOSS a preferred option for government procurement.

37. See *www.eweek.com/article2/0,3959,840669,00.asp for more details.*

38. More precisely, consumption of a non-rival good by one consumer does not decrease its utility for another consumer. Non-excludability implies that it is difficult, if not impossible, to charge people money for the use of the good, much as for breathing air or walking through a public park. Public goods are those that satisfy both the criteria of non-rivalry and non-excludability.

39. Full results are at *www.psychologie.uni-kiel.de/GNU/Linux-study/.* The three most important gains (all scoring 4.6 on a scale of 1 (very unimportant) to 5 (very important)) were "having fun programming", "improving my programming skills", and "facilitating my daily work due to better software." "Lack of payment" was much less important (2.2); "time lost due to my involvement in GNU/Linux was a bit more important" (2.6).

40. The Boston Consulting Group Hacker Survey, Release 0.3. was presented at GNU/LinuxWorld on 31 January 2002; *www.bcg.com/opensource/BCGHACKERSURVEY.pdf* . BCG surveyed a random selection of developers from SourceForge; the results are based on 526 respondents (a 34.2% response rate).

41. See Berlecon/III (2002), Part 4.

42. Any participating developer can express an opinion by casting a vote on any issue facing the project, but only the votes of Apache Group members are binding. Code changes require a minimum of three positive votes and no negative votes; vetoes are expected to carry with them a convincing explanation. Other decisions require a minimum of three positive votes and an overall majority in favor. Election to the Apache Group is on the principle of a peer-based meritocracy: someone who does a lot of good work on a piece of the code may be nominated by a member of the group and added to the group by a unanimous vote of existing members. Interview with Apache Group members; Fielding (1999).

43. For more details browse *www.apache.org.*

44. For more on the South African position see Government IT Officers Council of South Africa (2002), *Using Open Source Software in Government*; and National Advisory Council on Innovation of South Africa (2002), *Open Software and Open Standards in South Africa.*

45. See *http://tdil.mit.gov.in/* with a link to Indix (the Hindi version of GNU/Linux); see also *www.crn-india.com/features/stories/39090.html* and *www.zdnetindia.com/techzone/linuxcentre/stories/70365.html.*

46. See *www.redflag-linux.com/.*

47. For more information see *www.pernambuco.com/tecnologia/arquivo/softlivre1.html.*

48. See *www.wikipedia.org/wiki/Open_content* for a list of open content projects and links.

49. See *www.lightandmatter.com/article/article.html.*

50. See *www.sanger.ac.uk/HGP/.*

51. See *www.oreillynet.com/pub/a/network/2002/04/05/kent.html* and *www.wired.com/news/medtech/0,1286,46154,00.html* for more details.

52. See *http://bioinformatics.org/.*

53. See *www.newamerica.net/index.cfm?pg=article&pubID=901* and *www.cellularsignaling.org/.*

54. Programming that secretly gathers information about a computer's user and sends it to advertisers or other interested parties.

55. For the detailed text see *www.theregister.co.uk/content/4/25157.html* and *www.pimientolinux.com/peru2ms/.*

56. See Bar F and Borrus M (1998), The path not yet taken: User-driven innovation and U.S. telecommunications policy, Fourth Annual CRTPS Conference, University of Michigan Business School, Ann Arbor, Michigan, 5–6 June.

57. For more information see the Extreamadura FOSS site *www.linex.org* or refer to *The Washington Post* (2002), Europe's Microsoft alternative: Region in Spain abandons windows, embraces Linux (3 November) and *Wired,* Extremadura measures: Linux, at *www.wired.com/news/business/0,1367,51994,00.html.*

58. Office of the E-Envoy, Open Source Software Use in UK Government, *www.e-envoy.gov.uk/oee/oee.nsf/sections/frameworks-oss-policy/$file/oss-policy.htm.*

59. See *www.fossfa.org.*

60. For more details see ZDNet at *http://news.zdnet.co.uk/story/0,,t269-s2121266,00.html.*

61. See *www.lugcos.org.ar/serv/mirrors/proposicion/proyecto/leyes/#ref.#1.*

62. See *www.softwarelivre.org/index.php?menu=projeto* and *www.pernambuco.com/tecnologia/arquivo/softlivre1.html.*

63. See *www.redflag-linux.com/eindex.html* and *www.bsw.gov.*cn.

64. See *www.zdnetindia.com/techzone/enterprise/stories/74137.html;*
www.simputer.org/simputer/;
http://rohini.ncst.ernet.in/indix/;

http://economictimes.indiatimes.com/cms.dll/xml/uncomp/articleshow?artid=24598339; www.zdnetindia.com/news/national/stories/71697.html; and *http://ebb.antville.org/stories/362705/.*

65. See *http://asia.cnet.com/newstech/systems/0,39001153,39071821,00.htm; http://star-techcentral.com/tech/story.asp?file=/2002/9/9/technology/09oss&sec=technology; www.mncc.com.my/oscc/oscc-main.html;* and *http://opensource.mimos.my/.*

66. See *www.tremu.gov.pk/task/Linux.htm.*

67. See *http://odfi.org/archives/000004.html#4.*

68. See *http://bayanihan.asti.dost.gov.ph/.*

69. See *http://en.hancom.com/index.html.*

70. See *www.oss.gov.za/.*

71. See *www.nectec.or.th/linux-sis/.*

72. See *www.idg.com.sg/idgwww.nsf/unidlookup/21744381DA98B64148256CA80007772E?OpenDocument.*

References and Bibliography

Berinato S (1999). Catering to the GNU/Linux Appetite. *PC Week*, 7 June: 103.

Berinato S (2000). GNU/Linux Graduates to Mainframes. *Industry Standard,* 17 May.

Berlecon Research and the International Institute of Infonomics (III), University of Maastricht (2002). *Free/Libre and Open Source Software: Survey and Study.* http://www.infonomics.nl/FLOSS

Bessen J (2002). What good is free software? In: Hahn R, ed. (2002). *Government Policy toward Open Source Software.* Washington, DC, AEI-Brookings Joint Center for Regulatory Studies.

Bessen J and Hunt R (2003). An empirical look at software patents. Research on Innovation. www.researchoninnovation.org

Business Week (2003). The GNU/Linux uprising. 3 March.

CNET (2000). IBM to join in GNU/Linux supercomputing effort. 21 March.

DeLamarter RT (1986). *Big Blue: IBM's Use and Abuse of Power.* New York, Dodd, Mead.

Evans SD (2002). Politics and programming: Government preferences for promoting open source software. In: Hahn R, ed. (2002). *Government Policy toward Open Source Software.* Washington, DC, AEI-Brookings Joint Center for Regulatory Studies.

Fielding RT (1999). Shared leadership in the Apache Project. *Communications of the ACM* 42 (2): 42–43.

Free Software Foundation) (FSF) (1991). *GNU General Public License, v. 2.0.* www.gnu.org/copyleft/gpl.html

Free Software Foundation (FSF) (1996). The free software definition. www.fsf.org/philosophy/free-sw.html

Ghosh RA (1998). Cooking pot markets: An economic model for the trade in free goods and services on the Internet. *First Monday* 3 (3).

Goldhaber MH (1997). The attention economy and the Net. *First Monday* 2 (4).

Holmström B (1999). Managerial incentive problems: A dynamic perspective. Working Paper 6875. Cambridge, MA, National Bureau of Economic Research.

Iannacci F (2002). The economics of open-source networks. *Communications & Strategies* 48. International Telecommunications Society.

Keats D (2003). Collaborative development of open content: A process model to unlock the potential for African universities. *First Monday* 8 (2). www.firstmonday.dk/issues/issue8_2/keats/

Kuan J (2003). Open source software as lead user's make or buy decision: A study of open and closed source quality. Paper presented at the second conference on "The Economics of the Software and Internet Industries", Toulouse, France, 17–18 January.

Lancashire D (2001). Code, culture and cash: The fading altruism of open source development. *First Monday* 6 (12).

Lerner J and Tirole J (2000). The simple economics of open source. Working Paper 7600. Cambridge, MA, National Bureau of Economic Research.

Lerner J and Tirole J (2001). The open source movement: Key research questions. *European Economic Review* 45.

Lessig L (2002). Open source baselines: Compared to what? In: Hahn R, ed. (2002). *Government Policy toward Open Source Software.* AEI-Brookings Joint Center for Regulatory Studies, Washington, DC.

Open Sources: Voices from the Open Source Revolution (1999). DiBona C, Ockman S and Stone M, eds. O'Reilly & Associates, Sebastopol, CA.

Pappas Johnson J (2001). Economics of open source software. F/OSS, Massachusetts Institute of Technology. http://opensource.mit.edu/

Raymond ES (1999a). The revenge of the hackers. In: *Open Sources: Voices from the Open Source Revolution*.

Raymond ES (1999b). The magic cauldron. At http://www.catb.org/~esr/writings/magic-cauldron/.

Raymond ES (2000). The cathedral and the bazaar. www.catb.org/~esr/writings/cathedral-bazaar/

Stallman R (1999). The GNU operating system and the free software movement. In: *Open Sources: Voices from the Open Source Revolution*.

Stallman R (2002). Free as in freedom. Ongoing. www.oreilly.com/openbook/freedom/

Tuomi I (2002). The lives and death of Moore's Law. *First Monday* 7 (11).

Weber S (2000). The political economy of open source software. Working Paper 140. Berkeley Round Table on the Information Economy. http://brie.berkeley.edu/~briewww/research/workingpapers.htm

ANNEX I

The text of the GNU General Public License is reproduced here in the form in which it appeared on the Free Software Foundation site http://www.gnu.org/licenses/gpl.txt on 13 August 2003.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software – to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.> Copyright©
> <year> <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright© year name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> <signature of Ty Coon>, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Annex II

The text of the Open Source Definition is reproduced here in the form in which it appeared on the Open Source Initiative site http://www.opensource.org/docs/definition.php on 13 August 2003.

The Open Source Definition

Version 1.9

The indented, italicized sections below appear as annotations to the Open Source Definition (OSD) and are ***not*** a part of the OSD.

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

> ***Rationale:*** *By constraining the license to require free redistribution, we eliminate the temptation to throw away many long-term gains in order to make a few short-term sales dollars. If we didn't do this, there would be lots of pressure for cooperators to defect.*

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost–preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

> ***Rationale:*** *We require access to un-obfuscated source code because you can't evolve programs without modifying them. Since our purpose is to make evolution easy, we require that modification be made easy.*

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

> ***Rationale:*** *The mere ability to read source isn't enough to support independent peer review and rapid evolutionary selection. For rapid evolution to happen, people need to be able to experiment with and redistribute modifications.*

4. Integrity of the Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

> ***Rationale:*** *Encouraging lots of improvement is a good thing, but users have a right to know who is responsible for the software they are using. Authors and maintainers have reciprocal right to know what they're being asked to support and protect their reputations.*
>
> *Accordingly, an open-source license must guarantee that source be readily available, but may require that it be distributed as pristine base sources plus patches. In this way, "unofficial" changes can be made available but readily distinguished from the base source.*

5. No Discrimination against Persons or Groups

The license must not discriminate against any person or group of persons.

> ***Rationale:*** *In order to get the maximum benefit from the process, the maximum diversity of persons and groups should be equally eligible to contribute to open sources. Therefore we forbid any open-source license from locking anybody out of the process.*
>
> *Some countries, including the United States, have export restrictions for certain types of software. An OSD-conformant license may warn licensees of applicable restrictions and remind them that they are obliged to obey the law; however, it may not incorporate such restrictions itself.*

6. No Discrimination against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

> ***Rationale:*** *The major intention of this clause is to prohibit license traps that prevent open source from being used commercially. We want commercial users to join our community, not feel excluded from it.*

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

> ***Rationale:*** *This clause is intended to forbid closing up software by indirect means such as requiring a non-disclosure agreement.*

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

> ***Rationale:*** *This clause forecloses yet another class of license traps.*

9. The License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

Rationale: *Distributors of open-source software have the right to make their own choices about their own software.*

Yes, the GPL is conformant with this requirement. Software linked with GPLed libraries only inherits the GPL if it forms a single work, not any software with which they are merely distributed.

10. The License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

Rationale: This provision is aimed specifically aimed at licenses which require an explicit gesture of assent in order to establish a contract between licensor and licensee. Provisions mandating so-called "click-wrap" may conflict with important methods of software distribution such as FTP download, CD-ROM anthologies, and Web mirroring; such provisions may also hinder code re-use. Conformant licenses must allow for the possibility that (a) redistribution of the software will take place over non-Web channels that do not support click-wrapping of the download, and that (b) the covered code (or re-used portions of covered code) may run in a non-GUI environment that cannot support popup dialogues.

Annex III

Statement of the Free and Open Source Software Foundation for Africa (FOSSFA)

The text is reproduced here in the form in which it appeared at http://www.prepcom.net/wsis/1046170300 on 13 August 2003.

Preamble

The potential of open source will improve productivity and quality of life in developing countries. The process of transformation into information societies requires the full participation of all member states.

Vision

Our vision is to promote sustainable, viable and cost-effective software products for Africa through education and local capacity building.

Principles

Africa should investigate how to leverage the opportunities presented by the emergence of open-source software in the context of limited financial resources and expertise.

Specifics

Africa can bridge the "digital divide" by adopting open source, thus narrowing the effect of techno-colonialism.

Plan of action

It is envisaged FOSSFA, in partnership with Governments, intergovernmental organizations, civil societies and other stakeholders, will spearhead initiatives that build skills through education and empowerment of women and youth.

Lobby all stakeholders to adopt open source as the platform to engineer solutions that meet the needs of the people.

Strategies

FOSSFA will:

iii. Create an awareness of free software and open source in Africa.
ii. Build capacity in free software and open source.
iii. Develop a knowledge warehouse of expertise in Africa.
iv. Develop the African Open Source Portal.

We intend to achieve these by:

i. Lobbying key organs such as Africa Union, UNECA, UNDP, Agence la Francophonie and NEPAD among others to support open-source development in Africa.
ii. Leveraging various free and open-source capacities and resources in Africa.
iii. Lobbying donor governments and other institutions to tie ICT funding to free and open-source software.
iv. Lobbying African governments to adopt free and open-source software.
v. Promoting open-source capacity and skill development in Africa through education with emphasis on women and youth.