Linux Adoption in the Public Sector: An Economic Analysis

Hal R. Varian and Carl Shapiro*

01 December 2003

1. Introduction, Findings, and Recommendations

The Linux operating system offers information technology managers in both the private and public sectors an increasingly attractive option as a computing platform to run the powerful computer servers that are at the heart of computer networks, including the Internet itself.[1] Platform software adoption decisions typically have lasting implications for subsequent adoption of *application* software as well as additional platform software itself.[2] The significance of the Linux adoption decision is further magnified, and made more complex, by the fact that Linux is *open source software*, in contrast to *commercial software*. On top of all that, widespread public-sector adoption of open source platform software can greatly affect the economic development of a country's entire software industry, a critically important consideration for public-sector decision makers.

We have been studying the economics of computer software markets for some twenty years, and are the authors of a widely-read book on the information economy, *Information Rules: A Strategic Guide to the Network Economy*. Given the complexity and importance of the decision whether to adopt the Linux operating system, we believe that an accessible discussion of the

---

* Hal R. Varian is the Class of 1944 Professor in the School of Information Management and Systems, the Haas School of Business, and the Department of Economics at the University of California at Berkeley. Carl Shapiro is the Transamerica Professor of Business Strategy at the Haas School of Business, U.C. Berkeley, and a Senior Consultant at Charles River Associates. We thank Bruce Abramson, Principal at Charles River Associates, and Madeline Schnapp, Research Analyst at Pegasus Engineering and Research, Inc., for extensive support in the preparation of this paper. We also thank IBM for financial support for this research.

[1] Server computers, "serve up" information or services to end users and generally operate without continuous human supervision. Server hardware may take the form of mainframes, workstations, and personal computers, with the choice depending on the scale and operating requirements of the job. Desktop computers, in contrast, are the common "personal computers" used for a variety of interactive tasks such as word processing, calculating, email, and web browsing.

[2] "Platform software," which includes operating systems, is software that offers various services to applications software, which is written to run on top of the platform software.

costs and benefits of adopting Linux, rather than a proprietary version of Unix or Windows, rooted in proven economic principles regarding software markets, will be helpful to public-sector decision makers. This paper is our contribution to that discussion.

Our principal findings and recommendations are as follows:

- The Linux operating system has achieved a "critical mass" sufficient to assure users that it will be available and improved for years to come, reducing the risk to users and to software developers of making investments associated with Linux.

- The Linux operating system has a number of very attractive features for information technology managers in both the private and public sectors: users adopting Linux are less likely to face "lock-in" than those adopting proprietary platform software, and they retain greater control over their own computing environments. These benefits are especially salient in complex computing environments where large users benefit from the ability to customize their software environment, as often occurs in the public sector.

- Open source software, such as Linux, typically uses open interfaces. Some commercial software uses open interfaces, some uses proprietary interfaces. Open interfaces typically lead to a larger, more robust, and more innovative industry and therefore software with open interfaces should be preferred by public sector officials, as long as it offers comparable quality to proprietary alternatives.

- Because Linux is open source *platform* software, adoption of Linux can help spur the development of a country's software sector, in part by promoting the training of programmers that enables them to develop applications that run on the Linux platform The adoption of the Linux platform may well promote the economic development of *commercial* software to run in that environment..

- Fears that the licensing terms associated with Linux discourage the development of commercial software are misplaced. The fact that Linux is open source software in no way requires that the development of application software running on Linux follow an open source model. Rather, we expect mixed computing environments – involving open source software and commercial software, that employ both open and proprietary interfaces – to flourish in the years ahead.

While focused on Linux, our discussion necessarily ranges more broadly into the economics of software markets and the differences between the traditional model of development of commercial software and the open source software model used by Linux, Apache, and other popular open source software. Background information on the economics of software markets is provided in the Appendix to this paper; readers seeking to explore these concepts in greater

depth are encouraged to look at our book, *Information Rules: A Strategic Guide to the Information Economy* (www.inforules.com).

Section 2 of this paper defines and explains the key concepts of open source software, commercial software, and open and proprietary interfaces. We stress here that open source software inter-operates with, and complements commercial software. We also identify some of the underlying economic conditions that are favorable to the open source model of software development. Section 3 then applies these general principles to evaluate the benefits and costs of Linux adoption from the user's perspective; this analysis applies equally to private- and public-sector users. Section 4 addresses the specific question of the licensing terms applicable to Linux, including the "General Public License" or "GPL." Section 5 then briefly comments on some additional considerations that apply to public-sector decision makers, especially the impact of the platform software choice on the economic development of a country's software industry. A summary follows.

2. Open Source and Commercial Software

GNU/Linux is the leading example today of open source software.[3] Managers considering adopting Linux need to understand how the open source model of software development works, and how open source software complements commercial software. To aid in that understanding, we offer here a more general discussion of open source software, commercial software, and software interfaces. This material will serve as a useful foundation when we evaluate the benefits and costs of adopting Linux.

A. Open Source Software and Commercial Software

Open Source Software (OSS) is software for which the source code[4] is available to the public, enabling anyone to copy, modify and redistribute the source code. Access to the source code allows users or programmers to inspect and understand the underlying program; they can

---

[3] The software operating environment commonly referred to as "Linux" should more properly be called GNU/Linux since it is a combination of software from the Free Software Foundation (which developed GNU) and the software kernel developed by Linux Torvalds.

[4] Programmers write software in various computer languages such as Fortran, C, C++, and Java. The original format in which the software is created is called "source code." This source code is then compiled into an "executable," "binary," or "object code" format that runs on the computer.

even extend or modify the source code, subject to certain licensing restrictions discussed below. *Commercial Software*, by contrast, is software that is distributed under commercial license agreements, usually for a fee. While there are many different approaches to commercial software licensing, it is frequently the case that the user of commercial software does not receive the copyrighted software source code and typically cannot redistribute the software itself or extensions to that software. Companies that develop commercial software typically employ intellectual property protection to maintain tight control over the source code they develop.

B. Open and Proprietary Interfaces

Virtually no piece of computer software operates in isolation. Therefore, software interfaces – the methods by which one software program interacts with another, with users, and with hardware – are critical to software functionality and to users. Software interfaces are especially important in the complex computing environments that characterize government agencies and large commercial establishments.

We can distinguish three different sorts of software interface. Application Programmer Interfaces, commonly known as APIs, which describe how applications request services from the operating system, form an important category of interfaces. User interfaces, which describe how software appears to and interacts with a user, comprise another critical set of interfaces. Finally, document formats, which describe how applications store and interpret data are yet another type of software interface.

We distinguish between "open" and "proprietary" interfaces. An interface that is controlled by a single group and not available for everyone to use freely is called a *proprietary interface*. Many commercial software companies maintain proprietary interfaces; file formats are a common example. Alternatively, an interface that is fully described in publicly available documents and available for anyone to use freely is an *open interface*.[5] For example, HTML 4.01 is completely specified in publicly available documents such as

[5] An open interface is documented and freely available for use by everyone without restrictions from its authors. There are some gray areas, such as published interfaces controlled by a single group; published interfaces whose use is restricted; interfaces which are shared among a designated group, but not public; multiple interfaces, and so on. The pure cases described above are adequate for our purposes.

http://www.w3.org/TR/html401/ so it is an open interface. Interfaces associated with open source software are typically open since they are fully described in the source code.

Open interfaces offer many advantages to users and to software developers. They tend to increase consumer choice and promote competition. They also make it easier for various programs to retain compatibility with each other as they are improved over time. As we discuss at length in our book, software vendors as well as users can benefit when open interfaces are established, thereby avoiding a "standards war" between incompatible, proprietary interfaces.

C. Open Source Software Complements Commercial Software

By highlighting the fundamental differences between open source software and commercial software as models of software development, we do not mean to suggest that users must choose one or the other type of software to serve all of their computing needs. To the contrary, we believe strongly that open source software and commercial software can, and will, co-exist and complement each other in the years ahead. For example, a number of commercial software applications – such as IBM's WebSphere, databases programs sold by Oracle and by IBM, and many other applications – run on the Linux operating system. Likewise, open source software – including much of the GNU software – runs on Sun's Solaris operating system as well as on Microsoft's Windows operating system.

For precisely these reasons, a government agency adopting Linux is by no means precluded from also picking commercial software for many of its applications running on Linux. In fact, the transparency of the Linux source code and the open interfaces between Linux and application software are *benefist* for those seeking to develop commercial software applications running on Linux..

D. Who Creates Open Source Software and Why?

Managers familiar with commercial software may naturally wonder why developers devote resources to the creation and improvement of open source software, since such software is available for free. While "free" is an obvious benefit to users, sophisticated users recognize that investments in systems that rely on open source software are risky unless there is a viable business model that will ensure that the software will be improved and supported over time.

Naturally, just as many commercial software projects fail, not all open source projects prove durable and successful. However, we believe the evidence is now clear that the open source community supporting Linux is robust and that Linux is here to stay. We reach this conclusion after studying the conditions generally most conducive to open source software development as well as the actual adoption of Linux and the growing community of Linux programmers.

Most open source developers view their participation as serving their own business, professional, or mission development. Individual contributors demonstrate their skills; since the source code is available for inspection, it can serve as a signal of professional competence and lead to both an enhanced reputation and rewarding career opportunities. But open source software does not, and probably cannot in the long run, rely exclusively on individuals contributing their time and expertise to development projects. Many for-profit companies have found it worthwhile to invest resources in the development of open source software. An open source approach can reduce development costs and make it possible to receive a broader array of feedback and input earlier in the product development cycle. Even companies large enough to develop their own software – including IBM and Sun Microsystems – have embraced and invested in open source software. Ultimately, these companies have determined that they can earn a sufficient return on their investment to make open source software expenditures attractive. Returns can come in various forms, including internal use of the resulting software. But the biggest commercial attraction is the sale or licensing of complementary products or services: the prospect of earning revenues from applications software, hardware, and services can make investment in Linux.

E. How Can a Business Based on Open Source Software be Profitable?

Strong economic arguments and ample real-world evidence support the existence of viable open source business models. Economic viability is important for users who seek assurance that the software they adopt will be available, supported, and improved for years to come. Naturally, users care about the sustainability of whatever software they adopt, since it is costly to be stranded with software that is no longer supported and upgraded. Such concerns are especially great for platform software. For commercial software, the stability of the developing company is a key concern. For open source software, the future prospects of the software hinge on the stability of the project itself. Of course, as with commercial software, some open source projects

will succeed, and others will fail. But substantial real-world evidence contradicts any broad claim that open source software development is economically unsustainable.

Companies currently working with open source software provide the empirical support for the economic viability of the open source model of software development. Some of these companies build their distribution and service businesses by contributing to the open source community. These companies assemble collections of open source programs, bundle them together, and sell them as "distributions;" payment is thus received not for the software *per se*, but rather for the selection and assembly skill needed to compile a workable distribution. One can think of this as a business model based on the "assembly" or "aggregation" of freely available pieces into a valuable whole. Viewed this way, these companies are not unusual in economic terms: restaurants and florists assemble components that are widely available to create a finished product with substantial added value.

Furthermore, while software license fees account for a portion of these companies' revenues, the true business opportunities lie in follow-on documentation, support, service, and customization. Distributors and systems integrators who are active participants in the open source community offer their customers expertise and a demonstrated commitment to keeping their knowledge at the cutting edge of software development—a compelling credential that helps them attract business. In many ways, these open source firms' business model recasts software as a service industry, rather than as a products industry.

Several well-known technology companies with roots far outside open source software have also released source code for selected products—and have encouraged their employees to join the open source community. IBM, Netscape, Sun, RealNetworks, and others have become important contributors to open source development. For at least some of these firms' employees, then, regular salaries motivate participation in open source projects. In return, their employers gain some influence over the direction taken by an army of talented programmers that they do not have to pay. The companies also position themselves to gain true user feedback about their products. Open source developers who extend and enhance these products in useful ways can increase demand for a company's full product line, and generate additional service and support contracts. Open source licenses also minimize the chances that a competitor could introduce secret proprietary incompatibilities into an existing open source project.

F. Who Coordinates the Development of Open Source Software?

Some coordination is required among the various contributors to an open source project if the associated code is to be improved and to avoid "splintering" into multiple, incompatible versions. Many commentators have asserted that the multiple "flavors" of Unix have inhibited the development of Unix and have made Unix more vulnerable to competition from Windows in the markets for workstation and server operating systems. We regard such splintering as one of the most significant threats to open source software projects over time. We therefore advise users to look for factors that will discourage or prevent the emergence of important incompatibilities.

The most common arrangement is for an individual or small group to initiate the project which then grows into a community of developers which can then coordinate or maintain the expanding project. Open source projects with strong coordinating groups, such as Linux, should be well-placed to avoid splintering.

G. Open Source Successes: Linux and Apache

GNU/Linux (generally just called "Linux") is almost certainly the best-known open source software project. While Linux's availability as a free download complicates any estimate of its usage, some sources have cited numbers in the range of 18 million users; in any event, it has certainly exhibited the highest growth rate of any server operating system over the past few years.[6]

Linux's basic construction provides some useful insights into the nature of both open source development and the distribution business model. There is no single Linux program. Instead, the project consists of a collection of modules, each contributed by a different developer, proofed, reviewed, modified, and extended by other contributors, and eventually incorporated into an official release by Linus Torvalds. Some of these modules combine to form the "Linux kernel," the core part of the operating system that provides a computer's basic functionality. Anyone using Linux needs these modules, though users running Linux on different hardware platforms may need slightly different implementations. Other modules are optional. Because

---

[6] See the Linux Counter site and the IDC report 27521, "Worldwide Linux Operating Environments Forecast and Analysis, 2002-2006" for attempts to estimate the size of the Linux market.

different users may prefer systems optimized in different ways, Linux's basic modularity allows customization to suit many different user needs. Assembling a Linux system from its modules is akin to building a car from components—not the sort of task that most users would choose to undertake. The various distributions thus simplify the task by pre-selecting combinations likely to suit most users' needs. Customization and support services can refine the implementation even further to meet the specific needs of a business or government office.

Other open source projects define middleware and application modules that interact with Linux Commercial software developers also develop middleware and applications to run on Linux. Users considering Linux adoption thus have numerous choices to meet their needs.

Another well-known open source software package is the Apache web server. Apache had its origins in a public domain web server released by the National Center for Supercomputer Applications, a research center managed by the University of Illinois. A team of programmers who coordinated their activities over the Web managed the further development of Apache.[7] Hard data on the number of Apache servers is available through Netcraft's monthly automated survey of web servers. Its November 2002 survey of 35,686,907 web sites showed that 60.54 percent were running Apache. The Microsoft Web server was second in popularity, with a 28.89 percent market share. Apache's numbers were hardly a fluke; it has been used to run more than 55% of the Web's servers for over four years.[8]

Other important open source projects include FreeBSD (an operating system based on the Berkeley Standard Distribution of Unix), Open Office (a suite of software productivity applications including a word processor, spreadsheet, presentation tools, and so on), TEX (a professional-quality typesetting system), Mozilla (the open source version of Netscape's browser), Eclipse (a Web-based platform for tool integration), and Perl (a scripting language).[9]

---

[7] See the Apache FAQ (frequently asked questions) for background on the Apache project http://httpd.apache.org/docs/misc/FAQ.html

[8] See http://www.netcraft.com/survey/

[9] Information about these projects may be found on their websites: http://www.freebsd.org/, http://www.openoffice.org/, http://www.tug.org/, http://www.mozilla.org/, http://www.eclipse.org/, and http://www.perl.org/.

H. Who Uses Linux and Apache?

A variety of individuals and organizations use Linux and Apache.[10] As mentioned above, the Apache web server has by far the largest market share of web. Many educational institutions have found that access to source code eases training. For non-technology companies, the choice between open source and proprietary software is often guided by the same considerations that would dictate a choice among competing proprietary products—and many have chosen open source products.

In the technology arena, the development model and the openness of the Linux source code can be particularly attractive to companies who want to repackage, embed, use it to host specialized services, or create complementary products. Many large technology companies, such as Amazon, Google, and Yahoo, use Linux or BSD for their operations—and in particular to power their servers. These are all companies whose core businesses require near-perfect computer performance and reliability. Even minor disruptions can cost them dearly. They generally cite the stability of the systems, the ease of modification, the low cost, and the importance of maintaining control over their technology as their primary motivation.

US government agencies such as the Department of Defense and the National Security Agency also have adopted Linux. The NSA has even created a "Security Enhanced Linux" that they have made available on their web site for users who have significant security requirements.[11] There have been reports of successful adoption of Linux at the local and state levels as well.[12]

Open source software appears to be used widely in Europe. The *FLOSS Report* surveyed 1,452 companies and public institutions in Germany, Sweden and the UK.[13] It found that 395, or

---

[10] Furthermore, given the importance of network effects, the established installed base of Linux and Apache directly increases the value of these platforms for users, attracting even more users through the process of positive feedback that is common in network industries.

[11] See the MITRE report on "Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defense" and Security-Enhanced Linux page at the NSA for a description of this program. For a readable survey recent developments in the US and around the world, see "Linux in Government" and "Governments Push Open-Source Software"

[12] For one account, see "Largo Loves Linux More than Ever"

[13] See http://www.infonomics.nl/FLOSS/. This report offers a wealth of information about the use of open-source software use in Europe along with details about developer and user motivations.

27 percent, were using open source software or were planning to do so in the next year. In Germany alone, nearly 44% of the respondents were using or planned to use open source software. A much smaller survey of 66 public sector IT Managers in Europe found 63% of the respondents used some open source software in their operations.[14] Literally dozens of countries are actively considering adoption of open source.

I. Economic Conditions Conducive to Open Source Software

Users around the world stand to benefit as open source software competes against commercial software. We fully expect each type of software to be more successful in some areas and welcome this diversity as part of the process by which competition will unfold in the software sector. Linux in particular benefits from a set of favorable market:

- Linux has a proven track record for running large, reliable computer systems in a cost-effective manner.
- Many users value the flexibility they enjoy, and the control they retain, from using Linux rather than commercial software for their server operating systems.
- Linux already has a large installed base of users, which stimulates the supply of applications running on Linux as well as the supply of programmers familiar with Linux.
- Linux draws in the skills of a diverse and robust developer community, fueled by the fact that developers can gain status or recognition from participation in the Linux effort.
- Linux has the leadership and institutions necessary to prevent splintering and to establish a roadmap.
- Linux has strong support from major technology companies that stand to benefit by offering an integrated package that meets users' needs and/or selling complements to the Linux operating system. Leading examples include Red Hat offering enterprise platforms and services, Intel selling processors, and IBM selling servers and associated services.

---

[14] See Patrice-Emmanuel Schmitz, "Use of Open Source in Europe", http://www.cri74.org/actualites/articles/2001/usages.htm

3. Linux Adoption: Benefits and Costs to Users

We are now ready to systematically consider the benefits and costs to users of adopting the Linux operating system as a "platform" on which to build their computing environment. As noted above, a great many large organizations in both the private and public sectors have already adopted Linux. Here we provide information to help others decide whether this choice makes sense for them.

A. Total Cost of Ownership

Industry experts and consultants widely agree that users should consider the total cost of ownership (TCO) of a software package when making long-lasting software adoptions decisions. While not controversial in principle, determining the TCO is practice can be very complex indeed. To begin with, the purchase price of software, while easily measured, is only one component of the total cost of ownership (TCO). User training, maintenance, upgrades and technical support can contribute far more to the TCO than does the initial purchase price of software. Because many of these costs arise after the original software purchase, cost comparisons are only meaningful if they consider costs over a project's entire lifetime. In fact, it is often important to look beyond the lifetime of the specific project for which the initial adoption decision was made, since the data, training, and procedures adopted in one project often survive the project.

There have been several attempts to compare the TCO of Windows and of Linux in various computing environments.[15] In most of the studies the difference in TCO is on the order of 10 or 15 percent. This difference is not large; a 10 percent difference in TCO could easily be swamped by local conditions, random events, and other considerations. To a first approximation, it seems reasonable to suppose that neither of these two platforms has a striking advantage over the other in terms of conventional measures of TCO.[16]

---

[15] Two studies prepared at the request of IBM are Total Cost of Ownership for Linux in the Enterprise (IBM) and Linux v Windows Total Cost of Ownership Comparison (Cybersource). Microsoft commissioned IDC to prepare a report on TCO of Windows and Linux which has just been released; see "Study Shows Windows Server Costs Less Than Linux to Operate".

[16] This view appears to be widely held; see, for example, "Linux-Windows TCO contest 'a wash' for now"

One of the most important components of TCO is the labor cost of system administrators and support personnel. The studies cited above generally use wages based on the US market. These costs may be dramatically lower in countries with lower labor costs. Since the purchase price of Linux software is substantially lower than proprietary alternatives, the TCO---which includes both the price of the software and the labor costs necessary to support it---could be significantly lower in such countries.

B. Switching Costs

Users are also well advised to pay careful attention to switching costs when making adoption decisions. While the costs of switching *to* a new system are salient, the costs of (subsequently) switching *away from* that system are also very important. Users should be very wary of adopting a system that will be difficult to switch *away from* in the future, in part because the lock-in associated with using such a system will reduce their future bargaining power with their vendor.

Vendors always have some incentive to make it difficult for users to switch to alternatives, while the users will generally want to preserve their flexibility. From the user's viewpoint, it is particularly important to make sure that file formats, data, system calls, APIs, interfaces, communication standards, and the like are well enough documented that it is easy to move data and programs from one vendor to another. Clearly, open source software, with its open interfaces, offers an advantage to users over commercial software with proprietary interfaces in this important respect. Of course, commercial software can neutralize this advantage if it offers truly open interfaces. One of the benefits of supporting open source software in general, and Linux in particular, is the resulting pressure brought to bear on commercial software vendors to open their own interfaces, to the benefit of users.

It is also worth noting that the dominant player in an industry typically has an incentive to maintain control over its interface and make it difficult for other vendors to interoperate. Conversely, industry players who are not in a dominant position have very strong incentives to interoperate with the dominant player, and, for that matter, with each other.

C. Software Quality

There are several dimensions to software quality. Reliability, maintainability, usability, security, and flexibility are all important. Anyone contemplating an adoption decision must

weigh the relative importance of these factors in their own environment before selecting a system. The relationship between the alternative models of software development and any of these quality metrics is often easiest to explain by describing the relative merits of Windows and Linux. It's important to remember, though, that these systems are hardly "representative." They are two of the best software packages currently available, and play central, defining roles in computing environments. It is thus inappropriate to extrapolate directly from Linux to other open source projects, or from Windows to other software whose developers chose to keep its source code secret; comparisons of other pairings could lead to different conclusions. Nevertheless, certain comparisons can highlight the ways that the development models point towards different quality tradeoffs.

1. Reliability

The reliability of certain open source software programs has long been recognized. The fact that Linux powers large sites such as Google is testimony to its reputation for reliability. Windows reliability has improved significantly in recent releases, but it is still debatable whether it has reached the same level as Linux. In the *FLOSS Survey* of 1,452 European organizations, 83% of the respondents reported that "higher stability" was a very important or an important reason for adopting open source software.[17]

Recent Microsoft operating systems are reputed to be more reliable than earlier releases. But they still appear to suffer from an inherent architectural disadvantage compared to Linux: the lack of a truly modular design. In fact, some have argued that Linux's reliability is an outgrowth of its modular design. It is easier to replace parts of a modular software system without affecting the way other parts operate than it is to replace corresponding parts of a holistic, integrated system. Such modularity is virtually a requirement of systems developed by multiple programmers operating more-or-less independently—a factor inherent in the open source development model.

2. Maintainability

Maintainability refers to the ease of keeping a system updated and running. In the past, updating packages on Linux has been easier than on Windows because Linux machines use

standardized package management techniques to control complexity. All files for a given program are generally stored in a few, well-documented places. Most configuration files are text-based so that people can read them more easily. Again, recent releases of Windows have reportedly included improvements in system maintainability, so that now patches and updates can be applied almost automatically. In the *FLOSS Survey*, 60% of the respondents cited "operation and administration cost savings" as a very important or important reason to use Linux.

3. Usability

Usability testing is an inherently costly activity, and single-proprietor software platforms retain a long-standing edge over their open source competitors in this regard—particularly on the desktop. Microsoft has invested heavily in applications usability in recent years and many usability experts have praised Microsoft XP as a significant advance over previous systems. Unlike Windows, Linux allows the user to customize the user environment, so different users can use different environments. It is possible to configure standard Linux operating environments to look and operate a lot like Windows, making it relatively easy for users to migrate from one system to another. As graphical user interface technology continues to mature, "revolutionary" new features will become fewer and further in-between, thus giving competitors more time to respond to any significant advances. As a result, the usability of desktop Linux software is likely to continue to advance.[18] However, it is worth remembering that usability to end users is often a much greater concern when selecting desktop software to be used throughout an organization than when selecting server software to be used only by sophisticated IT professionals.

4. Security

There has been an ongoing debate about the relative security of Windows and Linux. Linux is inherently a multi-user system and has many built-in safeguards to manage user security. Since the source code of the system is available and many developers actually access it, it is much easier to detect bugs. This cuts two ways: it is easier both for attackers to detect bugs and for defenders to fix them. Once a bug is detected, verification of the problem is easier with open

---

[17] See http://www.infonomics.nl/FLOSS/.

[18] For a summary of the current state of affairs, see "Usability and Open Source Software"

source software, because anyone can inspect the source and analyze the bug. Furthermore, open source allows knowledgeable users to configure their own systems to eliminate common vulnerabilities. The NSA's "secure Linux," for example, strips away various kinds of functionality in order to emphasize security. Offering open source for a secure system is some assurance for potential users that there are no back doors or other security flaws.[19]

5. Flexibility

Open source software is flexible, in the sense that it can be customized or modified to specific needs. The ability to customize open source facilitates experimentation and adaptation, which has led to a considerable amount of "user innovation"[20].

One form of customization is the elimination of all parts of Linux other than those directed to a specific narrow task. In other words, *Linux can be made smaller*. Mindi Linux[21], for example, fits on single floppy disk and can be used for data recovery. Coyote Linux, Trinux, and the Linux Router Project also offer single floppy implementations of Linux that are optimized for various applications, such as networking. The various flavors of embedded Linux[22] are used for special purpose hardware such as cash registers, personal digital assistants (PDAs), personal video recorders, such as Tivo, MP3 players and the like.

Conversely, *Linux can be made bigger and more powerful*. With Linux, computers can be clustered together to build more power computational engines that can be used for a variety of purposes such as data mining, file serving, database serving, or web serving, to flight simulation, computer graphics rendering, weather modeling.[23] Recently Linux developers have been active

---

[19] Users of open source software are convinced that it has better security properties. In the *FLOSS Survey*, 75% of the respondents said "better access protection" was a very important or important reason for adopting open source software. See http://www.infonomics.nl/FLOSS/. Noted security expert Ross Anderson uses a theoretical model of software quality to argue that to the first order, open and closed software systems have the same level of reliability, See Anderson, Ross, "Security in Open and Closed Systems: the Dance of Boltzman, Coase, and Moore", http://www.cl.cam.ac.uk/ftp/users/rja14/toulouse.pdf.

[20] See Eric von Hippel "Open Source Software projects as user innovation networks", http://www.idei.asso.fr/Commun/Conferences/Internet/OSS2002/Papiers/VonHippel.pdf. For a much longer list of special purpose Linux distributions, see The Linux Distribution List [Special Purpose Distribution]. This list includes applications for special hardware requirements, for visually impaired users, for ISPs, for real time applications (such as device monitoring), for multimedia, and for a vast number of other needs.

[21] http://www.geocities.com/hugorabson/mindi_linux.html

[22] http://www.linuxdevices.com/

[23] http://lcic.org/

in implementing "grid computing", which allows organizations to harness computing power from large arrays of computer distributed over the Internet.[24]

Linux can also be made highly secure, where that is a requirement. We have already mentioned the NSA implementation of secure Linux, but this is just one of several projects to "harden" Linux. Other examples include Engarde Linux and Bastille Linux.[25]

Finally, *Linux can be localized*. It is no accident that many open source developers hail from outside the United States. They have often wanted applications that worked well in their native languages and were unable to find such applications in commercially available software … so they wrote their own open source versions.[26]

4. Linux and the "General Public License"

Open source software is distributed subject to a number of different licenses.[27] Open source licenses, like all licenses, describe the conditions that accompany the product being transferred, in this case, the source code. The very idea of having "conditions" accompanying a "free" good confuses some people, and opponents of open source software have done their best to amplify that confusion. But the truth is that open source licenses are not really that unusual.

Generally speaking, licenses associated with open source software give users greater rights than do licenses associated with proprietary software, although the specific terms vary from one software license to another. Indeed, for most users, open source license terms are more liberal than those that accompany proprietary software. Users who do not distribute software can do virtually anything that they want with it. They can use it throughout their organizations, copy it as many times as they wish for internal use, modify it, customize it, and use it with software developed under any model with either open or secret source code. These rights apply to home users, to companies, and to governments. None of the open source licenses place any serious

---

[24] The Linux Clustering Information Center provides details. See e.g., Linux at IBM - News.

[25] See http://www.linuxmagazine.com/2002-09/harden_list.htm to get an idea of what is involved to "harden" a Unix installation.

[26] For partial lists of localized implementations of Linux, see http://www.linuxselfhelp.com/cats/localization_language.html and http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/other-lang.html.

[27] For a partial list of popular open source licenses, see http://www.gnu.org/licenses/license-list.html. Note that this page, like many open source resources, has been translated into numerous languages.

restrictions on *users*.[28] They do, however, place some restrictions on software *distributors*. Organizations that distribute software to outsiders, and in particular those earning revenues by developing, selling, or distributing software, should make sure that they understand both open source licenses in general, and the ones that protect the particular open source programs that they have brought into their organizations. Of course, the same is true for commercial software.

The basic idea behind all open source licenses is that open source developers value the open source community. They would like other developers to join them, and to contribute to the growing body of open source code.[29]

The General Public License (GPL),[30] which governs much of the software included in typical GNU/Linux distributions, is designed to encourage sharing of enhancements to open source software. One controversial provision of the GPL, the "copyleft" provision, requires developers who modify code licensed under the GPL and who distribute their modified software to users outside their organizations, to license that modified software under the GPL. This requirement means that the source code to the modified software must be open and freely available. The GPL has never been tested in court, and legal scholars are split over the scope and enforceability of some of its more controversial provisions (including copyleft).[31] Developers who exercise standard due diligence are likely to encounter few difficulties using open source software.

While the GPL is probably the most widely discussed of the open source licenses, it is far from the only one. Most of the others not only do not contain copyleft clauses, but explicitly allow developers to incorporate open source code into products whose code they decide to keep proprietary. For example, the BSD license allows anyone to copy and use the source code as long as appropriate attribution is made to the original creators. Microsoft uses some BSD source code in Windows, and states this use in the Windows license—as required by the BSD license.

---

[28] The liberal terms that open source licenses grant to users can be seen from the philosophical statements issued by the major open source community groups. *See* http://www.gnu.org/philosophy/free-sw.html, or http://www.opensource.org/docs/definition.php.

[29] Many of the books written about open source software include collections of licenses as Appendices. *See e.g.*, Donald K. Rosenberg, *Open Source: The Unauthorized White Papers*, IDG Books, 2000.

[30] See http://www.gnu.org/licenses/gpl.html. The GPL Frequently Asked Questions are particularly helpful.

[31] All challenges raised so far (of which we are aware) have been resolved amicably, at least in part because open source licenses emerged from an academic "ethic" to preserve a sense of community, not to trap unwitting developers into opening their source code.

For virtually all organizations, though, the most important thing to remember about open source licenses is that their unusual terms apply *only* to those who develop and distribute software outside their organization. By and large, users can treat open source code as if it were their own—as long as they remember that additional obligations apply when they start distributing the software outside of their own organization.

5. Economic Development

The first duty of public officials making choices about software platforms is to choose the system that is best suited to the task at hand. As we have seen, there are many cases where performance, reliability, and security of Linux is equal or superior to that of proprietary alternatives. When two systems have similar suitability for a given task, open interfaces become an important consideration since they typically lower the cost of interconnection and reduce switching costs, making it less likely that the customer will become locked in to a single vendor. Open interfaces encourage third-party developers to create applications, add-ons, and complementary products. The benefits of such products to users, and even to entire countries are so great that virtually every software vendor wants to *claim* openness. But the important question to ask is whether they really have an incentive to deliver on this claim, not only now, but down the road when costs of switching to an alternative system could be very large.

Countries hoping to stimulate a strong domestic software industry should look first to their university system and ask "what environment will be most helpful in educating our future software developers?" Open interfaces are critical since they allow for local development of third-party applications. Open source platform software and commercial platform software with open interfaces both offer such opportunities to local software companies. In contrast, commercial platform software with proprietary interfaces can leave third-party developers at a strategic disadvantage relative to the company controlling the interface between the platform software and applications. Such a strategic dependency can discourage local investment of money and human resources in the development of commercial applications software.

Open source also plays an important role in that it exposes the inner workings of the software so that students can see just how quality software is put together. Just as aspiring auto mechanics need to actually work on real engines, aspiring systems engineers need to work on real operating

systems. Having such systems available, and open to scrutiny, will lead to better computer scientists, and better products in the future.

6. Summary

Open source software is here to stay. What was once a novel, even heretical, approach to software development has now been proven to work in practice. Viable business models exist for open source software developers, and users stand to benefit by selectively adopting open source software alongside commercial software.

Linux, in particular, has matured into a prime example of a successful open source project by developing durable institutions that enable compatible improvements to the Linux code. Many users in both the private and public sectors stand to benefit substantially from adopting Linux in their computing environments. Linux been proven to work well in the most demanding computing environments, offering an array of substantial advantages to many adopters: reliability, flexibility, security, and the avoidance of lock-in to a proprietary solution.

Public sector technology managers have additional reasons to adopt Linux. Adoption of Linux platform software promotes the training of software engineers and provides an open platform on which commercial or open source applications can be built, thereby spurring the development of a robust domestic industry. Certainly, any government information technology manager seeking to put in place a flexible computing environment that also helps promote the domestic software industry should give serious consideration to Linux.

Appendix: The Economics of Software Markets

The presence of strong network effects in platform markets, and the interrelationship between the platform and applications markets, make software a particularly complicated industry. Many of the economic effects that shape the industry's development lie closer to the cutting edge of modern economic thinking than to the basic theories taught in freshman courses. See our book, *Information Rules: A Strategic Guide to the Network Economy,* for a more complete discussion.

A number of these issues are particularly important: the complementarity among the components of an information system, the use of switching costs to lock in consumers and to guarantee revenue streams, the use of commitment as a negotiating tactic, the basic definition of network effects, the use of licensing terms to facilitate different business models, and the practice of bundling or integrating to increase revenues and alter market structures. The openness of source code can affect a number of these issues. While a review of these important aspects of network economics is beyond the scope of this paper, Table 1 summarizes some of the key economic effects shaping software markets.

ECONOMIC EFFECT	DEFINITION	IMPLICATION
Complementarity	The value of an operating system depends on availability of applications.	Consider the entire system of needs before making choice.
Switching costs	The cost of switching any one component of an IT system can be very high.	Make choices that preserve your flexibility in the future.
Commitment	Vendors may promise flexibility or low prices in the future but not deliver.	Look for firm commitments from vendors, such as a commitment to open interfaces.
Network effects	The value of an application or operating system may depend heavily on how many other users adopt it	For a closed network of users, standardization within the network is more important than choosing an industry standard.
Licensing terms	A perpetual license involves a one-time payment; a subscription involves a yearly payment	Licenses can be particularly pernicious when switching costs are high.
Bundling	Vendors will want to sell software in bundles to make future entry into the market difficult.	Purchasing a bundle now may reduce your future costs, but will also limit your flexibility and choices.

Table 1: Summary of Economic Effects and Their Implications

Policymakers need to appreciate that the decision to open source code is but part of a broader debate raging through the computer and software industries. Some in the industry have adopted the phrase "open computing" to describe an approach, applying to both hardware and software, that emphasizes modularity, interoperability, interconnectivity, and system flexibility. The key to open computing lies in open standards, including plug interfaces in hardware and application programming interfaces (APIs) in software. Important open source projects, such as Linux, embody all of these desiderata. Systems built around Linux are thus much better suited to the ideals of open computing than are systems built around platforms whose APIs are maintained as proprietary secrets. Many of the benefits that we attribute to open source software can be leveraged to even greater advantage when entire computing systems are open.

This is particularly true in considering the economics of openness. Because hardware and software, servers and desktops, platforms and applications, are all parts of a single computing environment, the economics of network effects ripple through the entire world of computing and software. For this reason, the effect of openness on industrial development are profound. Open standards and interoperability, in particular, tend to shift industrial focus from competition *for* the standard to competition *within* the standard. Immature industries may need time to experiment with different approaches before deciding upon a standard. Once competitors who began in different places converge, however, a standard exists—whether or not it has been "officially" recognized as such. If that standard remains the property of a single company, little competition may prevail. If, on the other hand, it is open to all industry participants, competition often remains fierce. Consumers and entrepreneurs tend to win, and rewards continue to flow to current innovators, rather than to those whose innovations proved successful during an earlier stage of industrial development.

Primer on Economics Concepts in the Software Sector

Complementarity

A decision-maker contemplating the adoption of a particular hardware or software platform must consider the entire information system. Hardware, software, personnel, training, system administration, and other components are all relevant to the adoption decision; looking at any one piece in isolation can be highly misleading. As a result, decisions about software adoption are more complicated than many other purchase decisions. Software adoption not only

influences decisions about hardware, training, and personnel, but also implicates concepts related to "lifecycle costing" and "network economics." Different models of software development, and differing degrees of access to source code, can change many of the calculations that should go into a thoughtful software adoption decision.

Switching Costs and Lock-In

Complementarity implies that the components of an information system are interdependent. Any decision to change a single component is likely to require changing others, as well. New hardware may require a new operating system. A new operating system may motivate new applications. New applications may require retraining. And new server software may necessitate updated desktop application software. These cascading changes impose "switching costs." When switching costs are large, users may be "locked in" to their current information system, or at least some components of it.

Huge switching costs and user lock-in both arise quite often in the world of information systems. Indeed, in many cases, the total cost to an organization of switching information systems vastly exceeds the purchase price of the hardware and software. When the costs of switching to an alternative system are large, and when the user must rely on a single vendor to provide components of the incumbent system (such as software or hardware upgrades), users may be locked in to a single incumbent vendor, and thus vulnerable to that vendor's whims—and more importantly, to its policies concerning service, support, licensing, and pricing.

Many information technology vendors rely on switching costs as an important part of their business models. Once a user has chosen a particular database vendor or an operating system, it may be very costly to change. This switching cost puts them at the mercy of the vendor. Savvy buyers should look not only at the deal that is offered up front, but also over the whole life cycle of the product. If the costs of switching to an alternative in the future will be very large, the locked-in consumer will possess little bargaining power. Consumers should always expect prices to increase for any future information technology services not included in their initial purchase contracts.

Decisions about information system adoption thus require consumers to "look ahead and reason back." Focusing only on the current situation can be quite misleading. Because

information systems are long-term investments that can lock consumers into vendors for many years, up-front decisions that maximize future flexibility convey true value to consumers.

Commitment

There is a fundamental tension between buyers and sellers when switching costs are large: buyers want to maintain flexibility while sellers want to encourage lock-in. Vendors recognize the reluctance of buyers to lock themselves in to proprietary solutions and thus try to downplay the extent of the lock-in. Open source alters the dynamics of these negotiations. It offers a way for sellers to commit not to exploit buyers after they have chosen an information system environment. If the source code for a software system is available, then users, perhaps aided by third parties, have the flexibility to maintain and to extend their own software investments. This ability allows users to adopt open source solutions with some degree of assurance that their switching costs will be relatively low. If they become unhappy with their current vendors, they can switch to others and have considerable control over their own switching costs. In short, low switching costs facilitate competition, thereby forcing vendors to stay on their toes and to provide good service after the initial sale is made. Customized software vendors have long used "source code escrow" to assure customers that they would not be stranded if the company went out of business. Open source is a much stronger assurance. It limits the extent of opportunistic behavior in the future and tends to produce a more competitive environment for vendors.

This effect, though, stems from an openness broader than simply open source code. Some parts of the industry use the phrase "open computing" to describe an approach, applying to both hardware and software, that emphasizes modularity, interoperability, interconnectivity, and system flexibility. The key to open computing lies in open standards, including plug interfaces in hardware and application programming interfaces (APIs) in software. Important open source projects, such as Linux, embody all of these desiderata; systems built around Linux are thus much easier to maintain as completely open computing systems than are those built around platforms with proprietary, closely-held interfaces. Many of the benefits that we attribute to open source software can be leveraged to even greater advantage when entire computing systems are open.

Network Effects and Positive Feedback

When the value of a product or service depends on how many other people adopt that product or service, economists say that there is a "network effect." For example, the value of a fax machine depends on how many other fax machines there are. Similarly the value of an email account may depend on how many of your correspondents use email.

In some cases, the value of a product may depend on how popular some other product is. A DVD player, for example, becomes more valuable as more DVD disks become available to play on it. When networks effects operate through complements in this fashion, they are known as "indirect network" effects. Computer software exhibits strong indirect network effects, since the value of an operating system depends, to some degree, on how many applications run on it. Similarly, the value of an application is enhanced if it runs on a popular operating system.

Such indirect network effects may be less important for a dedicated server—often what really matters is only whether a particular program, such as a Web server or database, runs on the server. Similarly, most users don't care what operating system is used in their cash register. But in other cases, a user might not know exactly what applications he or she wants when purchasing an operating system. In those cases, the system with the most available applications is attractive because it preserves options for the future; popular applications will be available, file exchanges will be easy, employees, customers, partners, or friends are likely to be familiar with the system, and so on. This inherent attractiveness born of sheer popularity means that the dominant operating system and dominant applications providers tend to have a large advantage compared to alternative providers, even when those alternatives are of similar quality.

Bundling

Software applications are often sold bundled together. Microsoft Windows itself consists of a large number of programs that work together; Microsoft Office involves several different "productivity tools" that interoperate. Red Hat Linux, a standard Linux distribution, involves hundreds of programs that all interoperate.

Bundling is an attractive policy for both vendors and buyers, though a specific bundle may serve the interests of only one party. Buyers may welcome a bundle because they can get a complete integrated package with some assurance that all the applications work together and that

they will be able to satisfy their future needs with this one package. Sellers may offer bundles since bundles allow them to better meet buyers' needs and perhaps to extend sales from one software category to another. Someone may initially buy Microsoft Office because she wants Microsoft Word. Later on, when she needs spreadsheet capability, she will naturally turn to Excel, which she already owns, rather that considering or purchasing competitive spreadsheet offerings. When these effects are strong, it may be extremely difficult for standalone vendors of individual software components, such as spreadsheets, to compete.

If there are switching costs associated with each component of the bundle, the cost of switching bundles will have to be summed across the various components. Even when each individual component has a manageable switching cost, the total summed across bundles may be substantial, leading to lock-in. Also, bundling may discourage users from switching one component at a time as a migration strategy.