Bernd Lutterbeck
Matthias Bärwolff
Robert A. Gehring (Hrsg.)
Open Source
Jahrbuch 2007
Zwischen freier Software und Gesellschaftsmodell

Linux Adoption in the Public Sector

HAL VARIAN AND CARL SHAPIRO

This article analyses the prospect of Linux adoption in the public sector. We conclude that the free operating system compares very favourable to common proprietary alternatives. Using Linux avoids the lock-in that typically results from opting for proprietary software, for its code as well as its interfaces are open. Using open source in the public sector also has positive effects on the software industry at large resulting in an overall improvement of a nation's economy.

Keywords: Public administration · Software quality · Open interfaces · Lock-in effects

1 Introduction

The Linux operating system offers information technology managers in both the private and public sector an increasingly attractive option as a computing platform, particularly to run powerful computer servers.[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 (OSS), in contrast to proprietary software. On top of all that, widespread public-sector adoption of open source platform software can greatly

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.

affect the economic development of a country's entire software industry, a critically important consideration for public-sector decision makers.

Given the complexity and importance of the decision whether to adopt the Linux operating system, we believe that an accessible discussion of the 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.

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 proprietary software and the OSS model used by Linux, *Apache*, and other popular open source software. For background information on the economics of software markets see our article in this volume (Varian & Shapiro 2007); readers seeking to explore these concepts in greater depth are encouraged to look at our book (Shapiro & Varian 1999).

This paper is structured as follows. First, we offer a brief outline of our principle findings and recommendations. Section 3 of this paper defines and explains the key concepts of OSS, proprietary software, and open and proprietary interfaces. We stress here that OSS inter-operates with, and complements proprietary software. We also identify some of the underlying economic conditions that are favorable to the open source model of software development. Section 4 goes on to apply 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. Last, we briefly make some additional considerations about the role of public-sector decision makers, especially with regard to the impact of the platform software choice on the economic development of a country's software industry.

2 Findings and Recommendations

Our principal findings and recommendations are as follows:

Critical Mass 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.

Attractive Features 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 Interfaces Open source software, such as Linux, typically uses open interfaces. Some proprietary 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.

Benefits Local SMEs[3] 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 proprietary software and services to run in that environment.

Best of Both Worlds 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 follows an open source model. Rather, we expect mixed computing environments—involving open source software and proprietary software, that employ both open and proprietary interfaces—to flourish in the years ahead.

3 Discussion of Open Source and Proprietary Software

GNU/Linux is the leading example today of OSS.[4] Managers considering adopting Linux need to understand how the open source model of software development works, and how OSS complements proprietary software. To aid in that understanding, we offer here a more general discussion of OSS, proprietary software, and software interfaces. This material will serve as a useful foundation when we evaluate the benefits and costs of adopting Linux.

3.1 Open Source and Proprietary Software

OSS is software for which the source code[5] is available to the public, enabling anyone to copy, modify and redistribute the source code. Access to the source code allows users

3 Small and Medium Size Enterprises (SME).

4 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 Linus Torvalds. GNU is an acronym for *GNU's Not Unix*.

5 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.

or programmers to inspect and understand the underlying program; they can even extend or modify the source code, subject to certain licensing restrictions discussed below.

Proprietary software, by contrast, is software that is distributed under proprietary license agreements, usually for a fee. While there are many different approaches to proprietary software licensing, it is frequently the case that the user of proprietary software does not receive the copyrighted software source code and typically cannot redistribute the software itself or extensions to that software. Companies that develop proprietary software typically employ intellectual property protection to maintain tight control over the source code they develop.

3.2 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 interfaces:

Application Programmer Interfaces, commonly known as *APIs*, which describe how applications request services from the operating system.

User interfaces, which describe how software appears to and interacts with a user.

Document formats, which describe how applications store and interpret data.

We may also 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.[6] For example, *Open Document Format (ODT)* is a standard for editable office documents completely specified by the *OASIS* industry consortium and freely available to the public while not being under exclusive control of but one company so it is an open interface.[7] Interfaces associated with OSS 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

6 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.

7 See http://www.oasis-open.org.

for various programs to retain compatibility with each other as they are improved over time. As we discuss at length in our book (Shapiro & Varian 1999), software vendors as well as users can benefit when open interfaces are established, thereby avoiding a "standards war" between incompatible, proprietary interfaces.

3.3 Complementarity between Open Source and Proprietary Software

By highlighting the fundamental differences between OSS and proprietary software as models of software development, we do not mean to suggest that users must chose one or the other type of software to serve all of their computing needs.

To the contrary, we believe strongly that OSS and proprietary software can, and will, co-exist and complement each other in the years ahead. For example, a number of proprietary 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, OSS—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 proprietary 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 benefits for those seeking to develop commercial software applications running on Linux.

3.4 Economic Conditions Conducive to Open Source Software

Users around the world stand to benefit as OSS competes against proprietary 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 conditions:

Proven Track Record Linux has a proven track record for running large, reliable computer systems in a cost-effective manner.

Flexibility and Control Many users value the flexibility they enjoy, and the control they retain, from using Linux rather than proprietary software for their server operating systems.

Broad Adoption 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.

Robust Community Linux draws on 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.

Governance Structures Linux has the leadership and institutions necessary to prevent splintering and to establish a roadmap.

Corporate Backing 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.

4 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.

4.1 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.[8] 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.

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

8 Two studies prepared at the request of *IBM*: Robert Francis Group (2002) and Cybersource (2002). See also a study by Bozman et al. (2002) commissioned by *Microsoft*.

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.

4.2 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 so that it is easy to move data and programs from one vendor to another.

Clearly, OSS, with its open interfaces, offers an advantage to users over proprietary software with proprietary interfaces in this important respect. Of course, proprietary software can neutralize this advantage if it offers truly open interfaces. One of the benefits of supporting OSS in general, and Linux in particular, is the resulting pressure brought to bear on proprietary 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.

4.3 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 is 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.

Reliability

The reliability of certain OSS programs has long been recognized. The fact that Linux powers large websites 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 OSS (Gosh et al. 2002).

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.

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 uses 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, however, 60 % of the respondents cited "operation and administration cost savings" as a very important or important reason to use Linux (Gosh et al. 2002).

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, but some high-profile open source desktop environment projects such as *GNOME* and *KDE* have also made significant progress over the last years.

Plus, Linux allows the user to customize the user environment extensively, 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 (Nichols & Twidale 2003).

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 OSS, 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 Security-Enhanced Linux (SE 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.[9]

Flexibility

OSS 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" (von Hippel 2002). Linux has been developed into a host of previously unforeseen directions.[10] In particular, the following aspects have received emphasis by various development and customization efforts:

Small One form of customization is the elimination of all parts of Linux other than those directed to a specific narrow task. *Mindi Linux*, for example, fits on single

9 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 (Gosh et al. 2002).

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 (Anderson 2002).

10 An impressive list of the huge variety of Linux distributions can be found at http://lwn.net/Distributions. The list includes distributions 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.

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[11] 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.

Bigger and More Powerful With Linux, computers can be clustered together to build more powerful 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. Recently Linux developers have been active in implementing *grid computing*, which allows organizations to harness computing power from large *arrays* of computers distributed over the Internet.[12]

Highly Secure 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*.

Localized It is no accident that open source is hugely popular with smaller language communities. They have often wanted applications that worked well in their native languages and were unable to find such applications in existing proprietary software offerings. Thus, they wrote their own open source versions.[13]

Linux allows for a level of freedom in customization and in-house development that is impractical to achieve with proprietary software. Thus, managers seeking flexibility and control are well advised to opt for Linux and build the in-house expertise needed to realise the full benefits of open source.

5 Open Source in Education and Economic Development

The first duty of public officials making choices about software platforms is to chose 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

11 For information on embedded Linux systems see http://www.linuxdevices.com.

12 The *Linux Clustering Information Center* (http://lcic.org) provides details.

13 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.

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 proprietary platform software with open interfaces both offer such opportunities to local software companies. In contrast, proprietary 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 proprietary 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 Conclusion

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 OSS developers, and users stand to benefit by selectively adopting OSS alongside proprietary 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 has 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 proprietary 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, and, indeed, open source software in general.

References

Anderson, R. (2002), Security in Open and Closed Systems: The Dance of Boltzman, Coase, and Moore, Working Paper, Cambridge University. http://www.cl.cam.ac.uk/ftp/users/rja14/toulouse.pdf [Jan 27, 2007].

Bozman, J., Gillen, A., Kolodgy, C., Kusnetzky, D., Perry, R. & Shiang, D. (2002), Windows 2000 Versus Linux in Enterprise Computing, IDC White Paper, IDC. http://www.microsoft.com/windows2000/docs/TCO.pdf [Jan 27, 2007].

Cybersource (2002), Linux vs. Windows: Total Cost of Ownership Comparison, Study, Cybersource. http://www.cyber.com.au/cyber/about/linux_vs_windows_tco_comparison.pdf [Jan 27, 2007].

Gosh, R., Krieger, B., Glott, R. & Robles, G. (2002), FLOSS – Free/Libre and Open Source Software, Survey and Study commissioned by the EU, International Institute of Infonomics, University of Maastricht und Berlecon Research GmbH. http://www.infonomics.nl/FLOSS/report/ [Jan 27, 2007].

Nichols, D. M. & Twidale, M. B. (2003), 'The Usability of Open Source Software', *First Monday* **8**(1). www.firstmonday.org/issues/issue8_1/nichols/ [Jan 27, 2007].

Robert Francis Group (2002), Total Cost of Ownership for Linux in the Enterprise, Study, Robert Francis Group. http://www.ibm.com/linux/RFG-LinuxTCO-vFINAL-Jul2002.pdf [Jan 27, 2007].

Shapiro, C. & Varian, H. (1999), *Information Rules: A Strategic Guide to the Network Economy*, Harvard Business School Press, Boston.

Varian, H. & Shapiro, C. (2007), The Economics of Software Markets, *in* B. Lutterbeck, M. Bärwolff & R. A. Gehring, eds, 'Open Source Jahrbuch 2007 – Zwischen freier Software und Gesellschaftsmodell', Lehmanns Media, Berlin. http://www.opensourcejahrbuch.de.

von Hippel, E. (2002), Open Source Software Projects as User Innovation Networks, Working Paper, MIT Sloan School of Management. http://idei.fr/doc/conf/sic/papers_2002/vonhippel.pdf [Jan 27, 2007].