Should You Adopt Open Source Software?

Kris Ven, Jan Verelst, and Herwig Mannaert, *University of Antwerp*

Organizations must consider the advantages and disadvantages of open source software before adopting it.

Many organizations use open source infrastructure software such as Linux, and open source software (OSS) is generally considered a viable technology. Both professional and academic literature devote much attention to the OSS phenomenon. However, decision makers considering the adoption of OSS face a plethora of books, research papers, and articles highlighting OSS's advantages and disadvantages. Different articles attach different levels of importance to these advantages or factors related to the adoption decision. Reasons for adopting OSS vary from the pragmatic to the ideological.[1,2] Some articles even contain seemingly incoherent or contradictory conclusions or advice. Consequently, managers are often uncertain about the criteria on which to base their adoption decisions.

Managers must take care when adopting OSS. Doing so for the wrong reasons can harm the organization, whereas not adopting OSS might leave considerable opportunities unused. Here, we provide a research-based, practical guide for interpreting evaluation criteria for OSS. This guide is firmly grounded in the available professional and academic literature, as well as in a case study involving several organizations. We distill important conclusions about the adoption of OSS that illustrate how decision makers can deal with these conflicting claims.

Research design

At the beginning of our study, we performed an extensive literature review on OSS adoption to gain important background information on potential adoption factors. Next, we conducted a case study to gain further insight into OSS adoption and to contrast our findings with the literature. Our main focus was infrastructure software, such as Linux and Apache. We selected 10 Belgian organizations from various sectors and of different sizes (measured by the number of employees). We included only organizations that used open source infrastructure software. In each organization, we conducted a face-to-face interview with key employees who were highly knowledgeable about the decision to use OSS. Typically, this included a manager (such as an IT manager) and a technical employee (such as a system administrator). The interviews aimed to gain insight into why the organization used OSS, as well as to obtain background information on its decision-making process. We recorded and transcribed each interview and asked follow-up questions by telephone or email. We analyzed the data using procedures to generate theory from qualitative data. We coded the interview transcripts for adoption factors and created a data display for each case. Each display gave an overview of applicable adoption factors, with corresponding quotes from the interview. During cross-case analysis, we merged the displays into one table.

0740-7459/08/$25.00 © 2008 IEEE

Table 1

Claims and counterclaims about open source software

Factor	Claims	Counterclaims
Cost advantage	■ OSS is free of charge[1–10] ■ Linux can lower hardware costs[2,4,8–10]	■ Enterprise Linux isn't free of charge ■ Dual licensing might require a commercial license ■ Unclear total cost of ownership[4,5,7,9] ■ Switching costs can be high[4,6,8,9,11,12]
Source code	■ Source code availability leads to higher quality;[1,3,9,13,14] enables customizations;[7,9,13,14] provides more choice and control;[9,13,14] and provides more trust in the software[1,7,9] ■ Source code can be important when developing products based on OSS because it provides more insight[7]	■ Lack of knowledge to apply modifications[2,5] ■ Lack of need to apply bug fixes[2,5,6] ■ Source code might not matter to organizations[2,4,5,13,14]
Maturity	■ Linux/OSS is reliable[2,4,8–10,12] ■ Category killers in horizontal domains[3]	■ Linux/OSS is unreliable[2,4,8,10,12]
Vendor lock-in	■ OSS avoids vendor lock-in[2,5,8–10,12]	■ Choice for enterprise Linux might be mandated by external vendor ■ Still dependent on OSS vendor for updates, services, and support
External support	■ External support is important for OSS[4–6,9,11,12] ■ Support for OSS is available from commercial vendors	■ Type of required support differs[9] ■ Support is lacking for some types of OSS[9]

Five contradictory claims

We identified five distinct adoption factors and evaluation criteria on which the literature makes contradictory claims and on which interviewees have various opinions. Other factors are also important in the adoption decision, but given the space available, we focus on those that stimulate the most discussion. Table 1 summarizes these factors.

Cost advantage

The OSS movement has always tried to downplay the fact that OSS generally doesn't require a license fee. On the other hand, all organizations in our case study indicated that lower cost helped drive the use of OSS. Many other studies have also shown that organizations tend to appreciate the fact that OSS is free of charge.[2–10] This perception might, however, be misleading. Not all OSS is free, so OSS might not be less expensive than proprietary software.

Several enterprise Linux distributions are available, such as Red Hat Enterprise Linux and SUSE Linux Enterprise Server. These products are based on freely available Linux distributions, but include additional services for enterprise customers, such as the certification of Linux for certain hardware, access to software updates, and support services. Some organizations are satisfied with a freely available Linux distribution, whereas others prefer the enterprise version. Organizations might base this choice on available in-house skills and the target system's strategic value. Some vendors might require the use of a commercial Linux distribution. SAP (from "Systems, Applications, and Products in Data Processing"), for example, only supports their products on Red Hat Enterprise Linux, SUSE Linux Enterprise Server, or Red Flag Advanced Server. So, an organization planning to install SAP on the Linux platform must buy one of these Linux distributions to obtain support from SAP.

Another situation in which the use of OSS might not be free is when using software from a vendor that uses a dual-licensing business model (for example, MySQL). Such vendors generally release their software under the terms of the GNU general public license. However, if an organization develops an application that incorporates software licensed under the GPL and starts to distribute it (for example, an application that uses MySQL as a database), the organization must publish that application's source code. Dual-licensing firms sell a commercial license for the same OSS product that doesn't require the application's source code to be licensed under the GNU GPL. The customer pays for the right to keep its intellectual property private.

To estimate the costs involved in introducing OSS, an organization can calculate the total cost of ownership. Various studies have compared the TCO of proprietary software with that of OSS, and many of these studies contradict each other. TCO studies should, however, be performed in the environment in which the adoption will occur because the result of one TCO study can't be generalized to other environments. None of the organizations in our sample but one performed a formal TCO calculation when considering OSS. They also didn't know whether the TCO would be beneficial for OSS. Other studies have obtained similar results.[4,5,7,9]

Switching costs are an important component of

OSS isn't always free, and the cost advantage over proprietary software might be limited, or even absent.

the TCO and occur when an organization moves away from the current platform and adopts a new one. They include the costs necessary to migrate data from the old system to the new and the costs required to retrain personnel. Although it's difficult to quantify such costs, employees' experience significantly influences them. This is most notable when considering the adoption of Linux. Six organizations in our sample indicated that it's much easier to migrate from Unix to Linux than from Microsoft Windows to Linux. This is consistent with other studies.[4,6,8,9,11,12] The easier transition is due to the fact that Linux is essentially a Unix clone so the two platforms share many administration tools. The Unix platform has always tried to use open standards, and OSS generally also strongly supports open standards. So, in most cases, you can migrate data relatively easily between the two platforms. Organizations using proprietary standards, however, might face significant costs during data migration. Hence, the installed base will largely determine whether OSS can easily be deployed within the organization.

Linux also lets some organizations lower their hardware costs. All organizations in our case study that were using Unix indicated that they did or might realize a significant reduction in hardware costs by replacing their proprietary Unix systems with Linux running on less expensive Intel hardware. Other studies have obtained similar results.[2,4,8–10] Only organizations that currently use Unix can realize this reduction in hardware cost because Microsoft Windows also runs on Intel hardware. In addition, the applications to be installed under the operating system must be compatible with Linux.

As we've illustrated, OSS isn't always free, and the cost advantage over proprietary software might be limited, or even absent. Therefore, OSS's potentially lower cost isn't necessarily a sufficient condition for adoption.

Source code availability

The OSS movement has always emphasized the advantages of source code availability. Proponents argue that making source code available lets everyone peer review the code, resulting in higher-quality software. It's also suggested that it gives users more choice and control because it lets them read and modify the source code. Although many OSS advocates have proclaimed these advantages, several authors have questioned or cast doubt on them.[3,13,14] In fact, you could argue both sides, depending on the situation. We distinguish between three scenarios.

In the first scenario, the source code's availability is neither an advantage nor a disadvantage for the organization. Half of the organizations in our sample indicated that they didn't consider source code availability to be an advantage and that they never used the source code. This is consistent with other studies in this field.[2,4–6] Joseph Feller and Brian Fitzgerald labeled this the "Berkeley Conundrum," which questions the importance of the source code's availability if no one actually uses it.[13] At least two factors can account for this observation:

- Our study focused on highly mature infrastructure software such as Linux and Apache. Organizations have little need to modify the source code of such applications.[2,5,6] Other types of OSS might give different results.
- Few—even experienced—programmers can modify the source code of mature software such as Linux and Apache.[2,5]

In this scenario, OSS serves as a black box, and its advantages and disadvantages are comparable to proprietary packaged software.[14]

In the second scenario, the organization considers the source code availability to be an advantage, but doesn't use it to study or customize the program. Some organizations in our case study expressed a greater trust in OSS because of the source code's availability. They felt that the program was less likely to contain hidden features and that bugs in the software would be quickly fixed. In addition, the source code arguably gives organizations more control over their IT infrastructure. An organization can access (portions of) the source code of proprietary applications, either through vendor programs (such as those of Microsoft and Oracle) or through escrow agreements. However, vendors generally limit the organization's rights. OSS gives organizations full access to the source code and generally places no restrictions on their right to modify or redistribute it. So, any interested party can obtain the source code and further develop, maintain, distribute, and support the software.

Although organizations in this scenario might not actually use the source code, its availability gives them the option of doing so later. In this respect, the use of OSS implies a learning process, in which the organization gains experience and skills in OSS. This process can, however, be a considerable investment. Depending on the modification's nature and the application's modularity, developers might have to study a considerable portion of the source code, even for a limited modification.

In the third scenario, OSS serves as a white box.[14] Organizations can use the source code to

study the software's inner workings or to adapt the software to their own needs. This is primarily interesting for organizations developing OSS-based applications. In this case, the source code availability can be an important factor in the adoption decision.[7,9] Three organizations in our sample indicated that although they didn't modify the OSS, the source code's availability let them better understand the OSS components' inner workings. This helped them locate errors in the software developed on top of OSS. Organizations might also customize the software. For example, two organizations in our sample customized their web mail applications.

Evaluation Models for OSS

Several frameworks are available for determining the maturity of open source software.

- Navica Open Source Maturity Model (OSMM), www.navicasoft.com/pages/osmm.htm
- Open Business Readiness Rating (OpenBRR), www.openbrr.org
- Cap Gemini Open Source Maturity Model, www.seriouslyopen.org
- Qualification and Selection of Open Source software (QSOS), www.qsos.org

Maturity

Another important question is whether OSS is mature enough for use in organizations. Several OSS evaluation frameworks (see the "Evaluation Models for OSS" sidebar) aim to help decision makers determine whether a particular OSS package is mature enough to adopt.

Reliability—that is, the software's ability to function as expected under certain conditions—is one aspect of maturity. OSS that receives a high maturity rating can be considered reliable. Claims concerning OSS's reliability go in both directions. However, making general comparisons in reliability between OSS and proprietary software is futile. Both cover a range of software, from extremely stable to rather unstable. Instead, organizations should make such comparisons at the product level. During our research, we encountered a considerable range of opinions concerning Linux's reliability. Some organizations still perceive Linux as being inferior to proprietary operating systems, whereas other organizations consider Linux to be as reliable as, or even more reliable than, Microsoft Windows and Unix. Other studies have found similar results.[2,4,8–10,12] This diversity in perceptions could indicate that operating system reliability depends on the environment in which it operates.

An organization's experience is also important when deciding whether an OSS package is mature enough to adopt. If an organization is unfamiliar with OSS, it should restrict its use to software that's generally considered to be mature. Examples of such OSS include several "category killers," such as Linux and Apache. Because OSS has traditionally been strong in horizontal domains,[3] the most mature types of OSS are Internet-based applications such as Apache or Sendmail.

In the past few years, much OSS development has been professionalized—that is, commercial software companies are increasingly investing money and manpower into the development of OSS projects. Several organizations we spoke to indicated that such evolutions further increased the trust in OSS. Moreover, many successful, mature OSS projects are backed by a commercial company or university. Using these applications generally represents a low risk for the organization because many other organizations have previously adopted them and much documentation and support is available. Apart from these OSS projects, thousands of projects that OSS maturity models would rate as less mature (for example, due to a limited number of developers) are mature enough for adoption. After the organization has familiarized itself with OSS and built up some experience, it might feel comfortable adopting such software. In that case, the organization must be able to assess the application's maturity and suitability in its specific environment.

Avoiding vendor lock-in

Organizations frequently adopt OSS to reduce vendor lock-in and become less dependent on their software vendors.[2,5,8–10,12] An organization that's locked-in to its current vendor depends on that vendor for its products and services, so switching vendors would entail significant cost. OSS can be considered an extension of the open systems movement, which mainly aims to ensure interoperability between Unix systems to reduce vendor lock-in. Researchers have argued that OSS support for open standards should facilitate the development of compatible products, avoiding dependence on a single vendor.[8] However, decision makers should be aware that although using OSS can reduce vendor lock-in, choosing OSS won't necessarily make them fully independent of vendors. For example, the organization might have limited choices among OSS products or vendors. As we mentioned earlier, vendors such as SAP require customers to use Linux enterprise versions.

Products offered by OSS companies such as SourceLabs and SpikeSource are another situation

Organizations shouldn't adopt OSS based on what other organizations do or on the various claims in the literature.

in which customers might be dependent on an OSS vendor. These vendors integrate several OSS components (such as Apache, Hibernate, Struts, and MySQL) into a certified, preconfigured OSS stack. The advantage of such stacks is that the components are fully integrated and well tested. Although the stack is freely available, customers requiring services for the OSS stack (for example, configuration and support) must contact the respective OSS company or one of its partners. Additionally, if a preconfigured OSS stack is used, the organization is tied to the specific version of the software the vendor uses. If upgrades or security patches become available for one of the OSS components in the stack, the organization must wait until the vendor has integrated these updates in its own stack.

A final example concerns the provision of OSS-related services. In principle, anyone can offer support for OSS products, which would increase the availability of support services for OSS. An organization would then be less dependent on a single vendor because it could choose among various vendors. However, commercially oriented OSS projects (such as Compiere and JBoss) prefer to approve official partners as insurance of high-quality support. These partners serve as a local contact point for support. Unfortunately, the availability of such partners is still limited in some countries. This limits organizations' choices and could make the organization somewhat dependent on the partner. For example, although Compiere has more than 100 official partners, in some countries only one or two partners are available. This situation should improve in the future, and Compiere will likely build an extensive ecosystem.

Although using OSS isn't likely to lead to vendor lock-in, organizations might not be as independent from their OSS vendor as would appear at first sight. Decision makers should therefore not adopt OSS simply to reduce their dependence on their vendor. Instead, they should investigate the degree to which their organization would depend on OSS vendors for services such as support or updates.

External support availability

The availability of external technical support has always influenced the adoption decision. All organizations in our sample considered the availability of external support for OSS to be important. Several other studies show similar results.[4–6,9,11,12] Support for OSS can take different forms. Some vendors offer support contracts—for example, the enterprise versions of Linux distributions and the services offered by companies such as SourceLabs. Large software vendors such as IBM openly declare their commitment to OSS and offer various services to customers. Additionally, many independent consultancy firms will install, configure, and maintain OSS systems.

Although research suggests that most organizations use some form of external OSS support, their reasons for doing so, and the type of support they choose, differ from organization to organization.[9] Organizations that haven't previously adopted OSS, or who currently lack the required skills, might prefer to outsource the installation, configuration, and maintenance. Several small organizations in our sample use an external consultant because their IT departments are understaffed and can't undertake the installation and support. Several other organizations use an enterprise version of Linux as insurance that if something goes wrong, the issue will be resolved in a reasonable amount of time. To select the appropriate type of support, an organization must be aware of its capabilities and requirements.

As we've shown, many claims and counterclaims exist with respect to the use of OSS. In addition, the interpretation of these factors is a complex activity that involves more than indicating whether a given proposition is true. The question then becomes how to interpret these conflicting claims when evaluating OSS. Our case study has shown that organizations tend to differ in their attitude toward certain advantages of OSS. This is consistent with previous studies.[2,5,8,9] As our analysis has shown, the context in which the organization uses OSS is critical in understanding these differences. For example, how organizations perceive the usefulness of the source code's availability will depend on how they use OSS (for example, whether they use it for development). We therefore argue that decision makers must consider the organization-specific context to assess the applicability and relevance of the various claims.

Organizations shouldn't adopt OSS based on what other organizations do or on the various claims in the literature. Instead, decision makers should carefully consider the organizational specifics before deciding whether to adopt. They shouldn't take widely claimed advantages—or disadvantages—of OSS for granted, but rather should investigate how each of these claims could manifest itself in an organization-specific context. Doing so will enable the organization to determine whether, when, and how to adopt OSS. Consequently, decision makers will need to construct their own rationale for adopting OSS. This attention to know-why and know-how influences the adoption's success.

Decision makers that critically examine the reasons for adopting a technology are said to be mindful in their decision making.[15]

A limitation of our research is that it focuses on Belgian organizations. Because OSS is a global phenomenon, cultural differences are likely to impact the adoption decision. The adoption of OSS in other regions might be subject to different expectations and results. So, decision makers should also pay attention to their local context when deciding whether to adopt OSS.

For organizations that are new to OSS, the conflicting reports might seem confusing and can be an obstacle to the technology's adoption. Such organizations should first adopt only mature OSS infrastructure software, such as Linux and Apache. These packages are well known and generally considered mature. Consequently, the conflicting evidence will be limited. Additionally, many commercial organizations offer support for this OSS, making it relatively easy to receive support. Once the organization has gained hands-on experience with OSS and is developing in-house knowledge on OSS, it might feel comfortable enough to investigate other OSS.

About the Authors

Kris Ven is a PhD student in the Department of Management Information Systems at the University of Antwerp. His main research interest are the organizational adoption of open source software and the link between OSS and innovation. Ven has a master's degree in management information systems from the University of Antwerp. Contact him at the Dept. of Management Information Systems, Univ. of Antwerp, Prinsstraat 13, B-2000 Antwerp, Belgium; kris.ven@ua.ac.be.

Jan Verelst is an associate professor in the Department of Management Information Systems and an executive professor in the Management School at the University of Antwerp. His research interests include conceptual modeling of information systems, evolvability and maintainability of information systems, empirical software engineering, and OSS. Verelst has a PhD in management information systems from the University of Antwerp. He is a member of the IEEE and the ACM. Contact him at the Dept. of Management Information Systems, Univ. of Antwerp, Prinsstraat 13, B-2000 Antwerp, Belgium; jan.verelst@ua.ac.be.

Herwig Mannaert is a professor in the Department of Management Information Systems and an executive professor in the Management School at the University of Antwerp. He is also cofounder of the software company Cast4All. His research interests include various aspects of software architectures, including design patterns and service-oriented architectures. Mannaert has a PhD in electrical engineering from the K.U. Leuven. Contact him at the Dept. of Management Information Systems, Univ. of Antwerp, Prinsstraat 13, B-2000 Antwerp, Belgium; herwig.mannaert@ua.ac.be.

References

1. K. Ven and J. Verelst, "The Impact of Ideology on the Organizational Adoption of Open Source Software," *J. Database Management*, vol. 19, no. 2, 2008, pp. 58–72.
2. J. Dedrick and J. West, "Movement Ideology vs. User Pragmatism in the Organizational Adoption of Open Source Software," *Computerization Movements and Technology Diffusion: From Mainframes to Ubiquitous Computing*, K.L. Kraemer and M. Elliott, eds., Information Today, 2007.
3. B. Fitzgerald and P.J. Agerfalk, "The Mysteries of Open Source Software: Black and White and Red All Over?" *Proc. 38th Hawaii Int'l Conf. System Sciences* (HICSS 05), IEEE CS Press, 2005, p. 196a.
4. J. Dedrick and J. West, "An Exploratory Study into Open Source Platform Adoption," *Proc. 37th Hawaii Int'l Conf. System Sciences* (HICSS 04), IEEE CS Press, 2004, p. 80265b.
5. M.H. Larsen, J. Holck, and M.K. Pedersen, "The Challenges of Open Source Software in IT Adoption," working paper 2004-11, Dept. of Informatics, Copenhagen Business School, 2004.
6. B. Fitzgerald and T. Kenny, "Open Source Software in the Trenches: Lessons from a Large Scale Implementation," *Proc. 24th Int'l Conf. Information Systems* (ICIS 03), Assoc. for Information Systems, 2003, pp. 316–326.
7. B. Lundell, B. Lings, and E. Lindqvist, "Perceptions and Uptake of Open Source in Swedish Organisations," *IFIP Int'l Federation for Information Processing*, vol. 203, E. Damiani et al., eds., Springer, 2006, pp. 155–163.
8. J. West and J. Dedrick, "Scope and Timing of Deployment: Moderators of Organizational Adoption of the Linux Server Platform," *Int'l J. IT Standards Research*, vol. 4, no. 2, 2006, pp. 1–23.
9. L. Morgan and P. Finnegan, "How Perceptions of Open Source Software Influence Adoption: An Exploratory Study," *Proc. 15th European Conf. Information Systems* (ECIS 07), Univ. of St. Gallen, 2007, pp. 973–984.
10. T. Wichmann, "Use of Open Source Software in Firms and Public Institutions—Evidence from Germany, Sweden and UK," Floss Final Report, pt. 1, Int'l Inst. of Infonomics, Berlecon Research, 2002.
11. S. Goode, "Something for Nothing: Management Rejection of Open Source Software in Australia's Top Firms," *Information & Management*, vol. 42, no. 5, 2005, pp. 669–681.
12. R. Ghosh and R. Glott, "Results and Policy Paper from Survey of Government Authorities," Free/Libre and Open Source Software: Policy Support (FlossPols) Deliverable D3, Merit, Univ. of Maastricht, 2005; www.flosspols.org/deliverables.php.
13. J. Feller and B. Fitzgerald, *Understanding Open Source Software Development*, Addison-Wesley, 2002.
14. C.B. Weinstock and S.A. Hissam, "Making Lightning Strike Twice," *Perspectives on Free and Open Source Software*, J. Feller et al., eds., MIT Press, 2005, pp. 143–159.
15. E.B. Swanson and N.C. Ramiller, "Innovating Mindfully with Information Technology," *MIS Quarterly*, vol. 28, no. 4, 2004, pp. 553–583.