UDC 004.428
DOI https://doi.org/10.32782/tnv-tech.2024.3.8

ADVANTAGES AND RISKS OF USING OPEN-SOURCE LIBRARIES IN COMMERCIAL DEVELOPMENT

Khambir V. R. *– Master,*
Principal Software Engineer, CapitalOne, USA
ORCID ID: 0009-0008-6812-2422

In the evolving landscape of digital libraries, the integration of Open-Source Software (OSS) presents a compelling avenue for enhancing service efficiency while curbing operational costs. This paper delves into the alignment of open-source principles with the foundational goals of libraries, underscoring the pivotal role of OSS in democratizing access to information and safeguarding intellectual property. Amidst the plethora of software options available for library management, the paper emphasizes the necessity of discerning the suitability of OSS for digital library applications, a decision critical to the continued growth and success of the open-source paradigm. The research extends beyond previous studies by offering a nuanced exploration of the functional and commercial benefits and drawbacks of OSS as perceived by management within the ESSS sector. It highlights the technical merits of OSS, such as enhanced reliability, security, and performance, attributed to the ability to scrutinize and modify the source code–advantages that notably surpass those offered by proprietary counterparts. The study also identifies parallel gains in the business domain, including the avoidance of vendor lock-in and the fostering of collaboration and innovation, which resonate with the technical benefits. The investigation reveals that certain technical challenges previously associated with OSS, such as compatibility issues and user-friendliness, do not pose significant obstacles for practitioners. Conversely, the study uncovers that the business disadvantages linked to OSS, although mirroring the technical concerns, present more substantial hurdles. In conclusion, the paper contributes a comprehensive assessment of the advantages and risks associated with the adoption of OSS in commercial development, offering valuable insights for practitioners and decision-makers in the realm of digital library services. It underscores the importance of balancing the technical and commercial considerations to harness the full potential of OSS, thereby shaping a future where open-source solutions are integral to the infrastructure of information access and preservation.

Key words: *Open-source software, Libraries, Commercial Development, Software Management.*

Хамбір В. Р. Переваги та ризики використання бібліотек із відкритим вихідним кодом (open-source) у комерційній розробці

В еволюції цифрових бібліотек інтеграція програмного забезпечення з відкритим вихідним кодом (OSS) представляє переконливий шлях для підвищення ефективності обслуговування при скороченні операційних витрат. У цій статті розглядається узгодження принципів відкритого коду з основними цілями бібліотек, підкреслюється ключова роль OSS у демократизації доступу до інформації та захисті інтелектуальної власності. Серед великої кількості варіантів програмного забезпечення, доступних для управління бібліотеками, у статті наголошується на необхідності визначення придатності OSS для програм цифрових бібліотек, рішення, яке має вирішальне значення для подальшого зростання та успіху парадигми відкритого коду. Дослідження виходить за рамки попередніх досліджень, пропонуючи детальне дослідження функціональних і комерційних переваг і недоліків OSS, як їх сприймає керівництво в секторі ESSS. У ньому висвітлюються технічні переваги OSS, такі як підвищена надійність, безпека та продуктивність, що пояснюється можливістю ретельного вивчення та модифікації вихідного коду – переваги, які значно перевершують ті, що пропонуються запатентованими аналогами. Дослідження також визначає паралельні переваги в бізнес-сфері, включаючи уникнення прив'язки до постачальника та сприяння співпраці та інноваціям, які перегукуються з технічними перевагами. Розслідування показує, що певні технічні проблеми, які раніше були пов'язані з OSS, такі як проблеми сумісності та зручності користувача, не становлять значних перешкод для практиків. Навпаки, дослідження показує, що недоліки бізнесу, пов'язані з OSS, хоча й відображають технічні проблеми, представляють більш значні перешкоди. Підсумовуючи, у статті міститься всебічна оцінка переваг і ризиків, пов'язаних із впровадженням

OSS у комерційну розробку, пропонуючи цінну інформацію для практиків і осіб, які приймають рішення у сфері цифрових бібліотечних послуг. Це підкреслює важливість збалансування технічних і комерційних міркувань для використання повного потенціалу OSS, тим самим формуючи майбутнє, де рішення з відкритим кодом є невід'ємною частиною інфраструктури доступу до інформації та її збереження.

Ключові слова: *програмне забезпечення з відкритим вихідним кодом, бібліотеки, комерційна розробка, управління програмним забезпеченням.*

Introduction. Open-Source Software (OSS) gives libraries efficient solutions to reduce the cost of their services. The remarkable similarities between the goals of open-source and libraries highlight the significance of using free and OSS in libraries. The libraries provides free services to all community members; it does not seek financial gain but rather to protect the intellectual property of literature owners and grant new rights to deserving authors. Additionally, it seeks to help people access information that is beyond their reach and that they would not otherwise be able to obtain. It is currently impossible to ignore the use of OSS in libraries. With so many software programs available to handle every aspect of libraries operations, it is important to determine which of these programs are appropriate for usage in digital libraries. Determining early success is critical to the open-source movement's sustained development and expansion [1].

Definitions. Let's define the terms for the discussion as follows:

"Open Source" refers to a style of software licensing in which the program's source code is usually made accessible to users free of charge, with conditions that permit addition, modification, and redistribution–though sometimes with certain limitations. A variety of organizations may offer the software's support, training, upgrades, and other services, increasingly through commercial agreements. Open-source software is frequently, though not always, created via teamwork, with several people contributing different parts of the finished product. Software companies are increasingly donating in-house projects and compensated programming time to the free software community [2].

"Commercial Software" refers to the paradigm in which software created by a business is normally licensed in object, binary, or executable code to a client (either personally or via channels) for a charge. The business frequently offers consumers the assistance, instruction, upgrades, and other services of a similar kind that they require in order to utilize the program effectively. The software's source code is typically not disseminated to everybody and may only be replicated or modified in accordance with the procedures specified in such agreements. However, it could be made readily accessible to specific users of the application under special licenses or other arrangements [3].

Every one of these software models has the potential to become a successful business plan for software firms and provide tangible benefits to clients. Businesses are rapidly figuring out how to accept both models and let them coexist as they are not mutually exclusive. Some proprietary operating systems platforms, for instance, have profited from open-source development by embracing an open-source strategy for the system's lower tiers (like device drivers) while maintaining private features for the higher tiers (like the user interface). With this strategy, more attention can be paid to the design of the more advanced elements, where innovation may benefit clients more broadly. On the other hand, some software developers have given their commercially produced software to the open-source industry so that open-source solutions may run across more platforms. The IT ecosystem has benefited greatly from increased rivalry and a greater variety of competitively priced servers and desktop platform options. Software companies concentrate on and significantly improve on new challenges such resolving security and dependability concerns, as software solutions continue to advance [4].

Understanding open-source and commercial software. Both commercial and OSS techniques have advantages and disadvantages of their own, and depending on the context in which they are used, they may provide users a variety of benefits with trade-offs. Commercial off-the-shelf software products have been on the market for a long time, providing users with a large range of computer features and enormous productivity improvements. Larger enterprises' complicated business running requirements, which generic off-the-shelf products might not be able to sufficiently fulfill, have also been satisfied by customized commercial software solutions. Commercial software providers are always working to create products that are user-friendly, highly functional, and responsive to client requests, Value for funds and reinforced by a services ecosystem in response to the needs of clients who might not be technologically inclined and prefer hassle-free problem solving. For many years, open-source license has been around, mostly at academic and research institutions. Due to the commercial support or corporate backing of OSS in the market, it has garnered more attention in the recent past [5].

Customers now have access to a large variety of software options and providers, even in markets where there have historically been few rival solutions. Because it permits them to freely copy, alter, and subsequently redistribute the source code, some people choose open source. People who desire to change the program source code are drawn to features like these, for instance, in environments where a great level of customization can be necessary or in educational environments where experimenting is the main goal [5].

Being involved in a software development community may help members form collaborations and exchange ideas, which can help them forge important connections with developers outside of their own organizations. Through official and informal sharing, developers of both commercial and open-source software strive to establish these communities. Open standards, which are not to be confused with open-source software, are adopted by both user and developer communities because they may quickly improve interoperability [3, 4].

Users of software today have more alternatives at their disposal. User settings frequently employ a combination of commercial and open-source software platforms and apps to satisfy various needs. The rivalry amongst software suppliers has increased the software industry's responsiveness to customer requirements, which ultimately benefits customers by offering them more alternatives and more affordable solutions. We will study commercial and OSS from three angles: development, licensing, and business in order to gain a deeper understanding of both models [4].

Business. Businesses are able to continue because they make money from what they do. Profit margins are the main metric used by shareholders to assess company success. Although the business strategies of companies that sell commercial and open-source software differ somewhat, both types of companies need to figure out how to generate steady income. The economic viability of developing software only for its own sake is questionable. Since commercial software companies rely on customers licensing their product, they concentrate on the features, functionality, and innovation of their software in order to fulfill the demands of their clients. When new software releases offer enhanced features, functionality, and value, customers buy the updated versions. This incentive creates a huge flow of funding for research and development into new software, which leads to increased productivity, decreased operating costs, and new learning resources [6].

The hardware and support services that open-source manufacturers bundle around open-source software and charge for are how they make money. For example, several

businesses sell OSS packaged with their server or personal computer hardware. The businesses offer this hardware and charge extra for the services they render to make their hardware and software compatible. A further illustration would be a system integrator that generates income by developing unique solutions for clients with pre-existing OSS as a foundation, and billing the clients for the time and materials needed to make the adjustments required to satisfy the particular needs of the user. An alternative business strategy involves making an open-source program available for free download and turning the user base into paying clients for a fully functional version. In some cases, combining development resources to support an emerging technology may offer an OSS provider rigid indirect revenue or advantages, such as through the sale of their hardware and/or commercial software that is offered in addition to the OSS [7].

Development. The methodology used in software development is another element that has historically set open-source and commercial software apart. This is accelerating in evolution and convergence as elements of one model are incorporated into the other. Traditionally, the major code development utilized by commercial software development teams has taken place inside the boundaries of a single business or unit. When it comes to open-source development, there is usually a framework in place to allow for the participation of several stakeholders. This duality is merging into a single developmental model. Commercial teams of software developers nowadays have created frameworks for working together to produce software with teams located all over the world.

Additionally, there are OSS solutions developed by a single business or programmers supported by for-profit companies. Only one or two major contributors keep these open-source systems up to date. The fundamental development process shared by both commercial and open-source development methodologies is iteration–design, standards, coding, testing, release, and feedback. A core group of programmers creates the application and distributes it to the community for early testing. After using the application, the beta testers notify the programmers of any flaws and suggested fixes. Before the application is made publicly available, the programmers make changes to the source code to address the issues found [8].

A suitable framework that supports the creation of software by several teams or contributors and their varied viewpoints can speed up innovation, optimization, vulnerability-fixing, and time to market, according to years of experience in the field. Such frameworks are used in many of today's software development projects by both commercial and open-source software development teams. Both the open-source and commercial development platforms help programmers advance their personal development and skill sets. For many years, fundamental ideas in computer science have been presented in textbooks. Since these texts are regularly updated, students have access to an abundance of published material.

Over a long time since applications was first written, traditional education techniques based on such literature have created proficient developers. The secret is not that students have access to source code that they can simply copy from, but rather that professors and educators are good in imparting significant concepts to students in a way that teaches them how to develop their own code to implement those concepts. When they tackle challenging or complicated challenges, skilled programmers can become recognized for their personal contribution to software development, regardless of whether they are using commercial or open-source software models [9].

Licensing. The licensing of software is the most fundamental distinction between both the open-source and commercial software models. Commercial software providers usually follow the conventional software licensing model, in which a consumer pays

a price to use the program. Generally, the license only allows the consumer to use, copy, or modify the program in accordance with its conditions. The freedom to alter and redistribute the program are among the common characteristics of OSS, which is made accessible under a range of license schemes. As with commercial software, the license agreement is based on the copyright included in the software. Permissions and rights are given with certain restrictions.

Generally speaking, these terms limit the software's future modifications and distribution options rather than demanding payment for the program. The Berkeley Software Distribution (BSD) License7 and the GNU General Public License (GPL)6 are the two main methods of licensing open-source software. All software derivatives and later iterations must be licensed and distributed under the same conditions as the original program under the GPL. The GPL-covered source code is perpetually covered by the GPL. The creators of the GPL intended for it to be perpetual, which limits the ways in which developers working on GPL software can create, distribute, or market goods utilizing GPL source code. Developers may also encounter other difficulties, such as figuring out if software created on a GPL platform for software qualifies as a derivative work covered by the GPL [10].

Open-source Digital libraries Software. "Linux is a cancer that adheres itself to anything it touches in the sense of intellectual property." Ballmer cited Linux as an example of a program that makes use of the GNU General Public License (GPL), which is owned by the Free Software Foundation. His major "issue" was that, if open-source software is utilized in the creation of new software, the GPL requires the software developer to make their code publicly available as well. Not to add that the GPL went into effect in 2007, which allayed Ballmer's worries about the updated version. Microsoft's aversion to open-source software reversed course in the 2010s and they began to support this new strategy. This is just one example, but since it concerns Microsoft's perspective, it's an important one. Further details on the increasing popularity of open-source software will be provided in the paper's following part. This will lead to a detailed explanation of the numerous benefits and drawbacks of adopting OSS at a business level. The firms that utilize open-source software and the circumstances in which using OSS makes sense are covered in the subsequent sections. The businesses included in this section are essential to the creation and upkeep of the OSS that drives modern commerce [11, 12].

Microsoft. As was previously indicated in the report, Microsoft was the firm that resisted open-source software the most, but they have now changed their stance and begun to support it. Microsoft had the most workers that contributed to GitHub projects compared to other companies in 2016. It now collaborates with other top open-source businesses like Red Hat. Several of its most well-known programs, such as the CNTK deep learning toolkit, TypeScript, Redis, Visual Studio Code, PowerShell Code, and .NET development tools, were also made available as open-source projects. Along with supporting Linux on its web-based computing service, it develops software across several platforms [13].

IBM. One of the main companies that contributed to the Linux kernel was IBM. It also established and contributed to several other open-source projects, including OpenWhisk, Project Intu, and LoopBack. Most recently, it published the WebSphere Liberty project under the Eclipse Public License. Additionally, IBM sponsors or is a member of several prominent open-source foundations, such as the OpenStack Foundation, the Apache Software Foundation, the Eclipse Foundation, and the Linux Foundation [14].

Intel. With a 12.9% contribution percentage to the Linux kernel in 2016, Intel was the corporation most actively involved in kernel development. It also sponsors and participates in a number of open-source foundations, including as the OpenStack Foundation, the Eclipse Foundation, and the Linux Foundation, just like IBM does [15].

Google has published more than 2000 open-source projects and made contributions to them. On the list of the top GitHub contributors in 2016, it was ranked sixth. Angular, which ranked fourth on the same list, is also owned by Google. Google has several well-known open-source projects, including TensorFlow, Android, Kubernetes, Dart, and Chromium [16].

Facebook. In 2016, Facebook rose to prominence as a provider of open-source hardware and software, with the second-highest GitHub contributor count. Its most well-known open-source initiatives are Relay, Flow, HHVM, and the JavaScript development tools for React and React-native. [8] 4.6 Docker with over 8 billion downloads, the Docker containerization technology has become one of the most popular open-source projects for business customers and has emerged as one of GitHub's most downloaded repositories. Docker software is particularly popular among firms employing agile and DevOps methodologies, and the company states, "On average, companies utilizing Docker report a 7X boost in the number of times they're capable of shipping software [17]."

Adobe. With more than 250 publicly accessible repositories on the GitHub site, Adobe has demonstrated its strong dedication to open-source. Developer tools such as the PhoneGap web design structure, the Brackets text editor, and the Topcoat CSS libraries are among its most well-known open-source products. Additionally, members of the Adobe team frequently contribute to several other open-source projects, including Flex, Felix, Apache Cordova, Gecko, Blink, and WebKit [12]**.**

Formulation of the problem. In the field of developing commercial software, the linkage to open-source libraries is commonly performed. This approach provides scores of advantages for depending on it, including, but not limited to: The reduction of the costs required for successful development; The ability to speed the development process; The availability of a remarkable number of innovations created by the community. But it came with a new set of problems such as security problems, license problems, and problem of dependency. This paper discusses that currently there is the problem of a lack of a clear understanding of the extent of using open-source components and corresponding opportunities and threats in various commercial projects.

This research aims at providing a critical review and assessment of the consequences associated with the implementation of open-source libraries in developing and creating business-oriented software products. The study thus seeks to embrace not only the advantages of adopting open-source libraries but also the disadvantages where by this detailed investigation will enhance the perception developers, project managers and decision makers will enjoy as they make decisions on the necessity of integration of open-source libraries into decisions. Therefore, the findings of the research will help in advancing the current knowledge on how to optimally implement open-source software without the bearing the negative consequences of free software, thus improving on the usage of open-source solutions by the commercial world.

Purpose of the study. Open-source libraries have been employed in commercial software development and this study aims at ascertaining the benefits realized together with the danger of engaging in such practices. It is supposed to evaluate advantages, like the lower cost of manufacturing and getting access to modern technologies, and threats, like the compromising of security and the violation of the licensing agreement. Thus, this study aims at looking at the effects on the development processes in order to

understand the recommendations and procedures for integration. These results will be of great value for the developers and managers who will be able to improve the strategic application of open-source resources. It is thus safe to say that this research benefits the existing literature on open-source software by being one of the few that links theory to practice within the commercial realm.

Research analysis. While earlier developers considered OSS elements to be a non-cost delivery method, it gradually evolved to the proposition where business organizations have to invest time into using them. Thus, it made the ease of software customization and the possibility of turning to community/commercial sources as a priority when it comes to comparing different software [18]. The routine scanning for license compliance with the SBoM for software is being integrated by the use of SPDX; instance, by Siemens AG [19], OSTG [20], the Linux Foundation [21]. Most studies that have been conducted in the last few years have addressed various phases in the adoption of OSS components.

On the other hand, the usage and the degree of companies' engagement with OSS components have grown higher, however, there are limited sources available describing the practices follow in companies to support the OSS components' adoption. Some of the more formalized schemes for evaluating OSS software described 10 years ago by Yılmaz et.al (2022), their contemporaries have now disappeared from the academic and practitioner discussion and more recent studies have revealed that trends regarding the attitudes of the businesses toward the OSS components are evolving [22]. The most important factors for developers were the flexibility of software modification, the presence of available support from either the community or from a paying source; the most important factor for their managers was commercial support. As for the other elements, which were considered less but still relevant important, these were quality, flexibility, maturity and reliability [18].

The idea of OSS component adoption is not as straightforward as looking for functionally suitable software as the case might imply [23]. Companies have to make additional decisions, for instance, the software licence of the component taking into account the licensing strategy of the business [23]. A current initiative to create such structure is the Linux Foundation's Open Chain project [24], which has developed some standard, including SPDX , which does allow for automated compliance checking, for instance [25].

Fendt and Jaeger (2019) and Harutyunyan et al. (2019) discuss the issue of the extensive large software product containing OSS licensed components. Fendt and Jaeger (2019) explain the case of Siemens AG in terms of integrating the tool chains for the license compliance checking into CI/CD. One consideration is that the procedure for clearances of the license or the determination of the licensing of source code rather than accepting the word of the packager is costly and in a rich SBoM has to be carried out only once per package [19, 26]. Following are the summaries of the problem by Riehle and Harutyunyan (2019), some solutions and some of the research questions that remain unanswered. Yes, automation can be used but solutions now are constrained hence more tools need to be created [26].

Main presentation: Advantages and Disadvantages of Open-source libraries

Both the advantages and disadvantages of OSSs are many. However, the benefits outweigh the disadvantages. The next sections address the advantages and disadvantages.

Advantages of OSS

OSS's have more benefits over proprietary software's. Some of the advantages are as follows:

• Error free software: Like in computational programming there are multiple chances that the software might crash or any other bugs that occur, it is always preferred to be given the source code of the program so that anyone can handle the occurrence of the errors. This is an advantage against the commercial software's where modifications are done by either only professional and we have to wait until they resolve the issue and come with a solution. The only course of action that a user can take here is to inform the developer about the problem, for these reasons OSS are more flexible and errors are more quickly handled than in Commercial software's [27].

• Availability of source code: One of the major components of the current procedure is the source code, and it cannot be utilized for commercial software's. In open-source, codes of source are available to all users by viewing while in the other types of software, it is only visible to a developer or a programmer [28].

• Modification and Redistribution: The most noticeable characteristic is, that not only the source code is delivered, but the source code can even be modified regarding our specifications. They can even be redistributed under the same conditions, and this would favor the future users [29].

• Security purposes: Even if these people do not know fundamental facts about software, they try to convince people that closed software are safer than OSS which is not true for people who know about OSS and its advantages over closed software. For security and merely for maintenance of the OSS it is mandatory to state your OSS with the license terms. And the same terms of usage and rules are provided if the given software is altered and redistributed [30].

• Customization: However, when it comes to active usage of software in an institutional framework, there is always a requirement of a person having a copy of a particular software. This is due to the fact that whenever we are handling commercial software we are at a disadvantage of having no individual modeling of the particular software. As with a point of view, we are always expected to call to the developer any time we want to make any changes to the software, which is time consuming and costly each and every time we get to consult the programmer. The advantage of OSS is that in utilizing OSS we are able to incorporate any language that we like which is not possible in commercial software [30].

• Avoiding Lock-in: It becomes costly high in any time when that organization is already using software then every time if it wants to opt for software then it becomes high cost and the organization is bounded or we can also say locked. To get bounded to software which was adopted for doing a job is not a deal of being satisfied with, in regards an institute. Software have their lifespan unless there is another one with some new flexibility feature in the market. OSSs do not contain such types of locks and the user can use any particular software when they desire [31].

• Costs: Normally OSS is free and in case of sometime training, support or maintenance charge is very minimal, in fact which is also incomprehensible by any small institute [32].

Disadvantages of OSS

There are very less disadvantages of OSS's. Some of them are mentioned as follows:

• Warranty validity: The warranty clauses are as follows but they are valid for certain conditions. For instance, if the customer experiences a problem with the code during modification, then what he or she is experiencing disqualifies the warranty sentences [33].

• No development guarantees: An element of uncertainty is the fact that in a given period there might not be any development at all. If the code is not in action, implementations of the software on the other hand will be primitive. As far as any user

does not compile the source codes and does not make any changes in it, no growth of a software can be seen [34].

• Performance: The commercial software's may be faster than the OSS's, because its is receiving more traffic than the commercial software's. Thus, the highly profiled companies give preferences to the commercial software's rather than the OSS's [34].

• Maintenance costs: As it mentioned always it is free software most of the time 90% of the cost shows that it is just for the Maintenance [35].

• Trademarks: As for the OSSs developed by a given company, there are some of them do not desire to eradicate their trademark. This is a sort of deceptive Danish end user who needs to alter it, and once more resell it without trademarks [36].

• Certifications: The clients nowadays are in a position to pay for brands instead of choosing things that may be cheap, because the focus is on quality not the price. That is a known fact that commercial software's are very costly ones, but they may contain more efficiency than free software's [37]**.**

Many of the advantages are the same as those reported in the literature, but some new information also emerged, such as the additional business functionality provided by OSS and the creation of de facto standards. Only two of the technical disadvantages of OSS that have been documented in the literature–compatibility problems and a lack of experience–are supported by the study's findings. It was shown, therefore, that the issue of lack of competence is typically more closely linked to an absence of knowledge about OSS. The main perceived disadvantages were found to be inadequate documentation, an excessive number of interfaces, limited functionality, and a deficiency of roadmaps (Tables 1-3).

Table 1

Technical Benefits of OSS

Reliability	The majority of literature listed reliability as one of the primary technological advantages in terms of high application availability and dependability.
Security	The majority of literature felt that OSS offers superior security since it is readily available, poses less of a risk from viruses, and prioritizes security during the product design process. Two businesses believed that OSS would not improve security
Quality	Regarding improved quality from peer reviews and the caliber of developers and testers, the majority of literature said that quality was beneficial. According to two companies, this was limited to high-end, established OSS programs like Linux.
Performance	Literature mentioned having good capacity and fast performance. Three have not yet seen more proof of OSS's effectiveness, and two were unsure if OSS outperformed proprietary.
Flexibility of Use	Advantageous to the majority of literature because it permits flexibility, personalization, experimentation, and alteration
Developer & Tester Base	Very advantageous for the majority since it guarantees that OSS is current and of high-quality software.
Compatibility	Many stated that because OSS is very interested in preserving formats for improved interoperability, it helps to ensure compatibility. The remainder had not observed any proof of this or thought it was not worthwhile.
Harmonization	Enhanced standardization of procedures and activities related to interoperability

Table 2

Business Benefits of OSS

Low Cost	When it came to lower license costs, software upgrades, virus protection, and the overall cost of the package – that is, the software plus service – half of the literature thought this was advantageous. The other half believed there was no advantage at cheap cost.
Flexibility by licenses	Most people believe to have a major influence on lowering capital expenditure in businesses
Escapes vendor lock-in	Extremely advantageous for the majority since it allows for independence from commercial sellers, a sense of control, and freedom of choice. Two businesses believed that OSS may also be affected by vendor lock-in.
Increases collaboration	Increased cooperation is advantageous to most parties since OSS makes it easier to develop new products, cooperate and share expertise, creates new avenues for collaboration, and allows businesses to share costs.
Encourages innovation	The majority concluded that having access to the source code promotes more creativity by generating ideas and technical innovation while also expanding avenues for innovation.
Extra business functionality	Advantageous as it makes it possible to maintain small teams, which enhances output and communication
De facto standards	Not the only business taking action. It would be advantageous to create a standard that enables the business to concentrate on its core competencies.

Table 3

Technical Drawbacks of OSS

Compatibility Issues	Not implicitly disadvantageous but some businesses involvement compatibility difficulties with present technology, skills and tasks
Lack of Expertise	While it's true that the typical lay employee lacks experience, this might also be due to a lack of knowledge of OSS.
Poor documentation	Outdated documentation or maybe lost during development
Proliferation of Interfaces	Various builds frequently make it difficult to decide which one to use.
Less Functionality	Integration level inferior to that of Microsoft
Lack of Roadmaps	Makes it challenging for businesses to identify a strategic direction for the great majority of their goods. The majority of items lack a strategic purpose.

Table 4

Business Drawbacks of OSS

Lack of support	The majority said there was not any security since there was no organization to support it or any kind of help
Lack of ownership	It is impossible to hold someone accountable or liable for issues
Access to the source code	The possibility that certain employees would feel uneasy about disclosing source code. Lack of understanding on this matter
Insufficient marketing	OSS is not owned by a single entity, nor is there a marketing budget, therefore word-of-mouth advertising is the main source of OSS.

Continuation of table 4

Investments for training	Four businesses stated that Linux required more training expenditures than Windows. On the other hand, it was discovered that e obtains superior quality OSS training.
Finding the right staff/ competencies	Finding employees and developing their skills to work using OSS apps may be challenging.

It was discovered that managers face more difficulties dealing with the commercial disadvantages listed in Table 4 than they do with their technical equivalents. For instance, the bulk of the enterprises regarded a lack of help as a serious disadvantage. Teams of technicians from a few of the businesses are available for internal support. But for many smaller companies, this isn't always an option.

The research paper under consideration presents the reader with information on how artificial intelligence and computer programs have affected translation. It poses a question of whether any of the existing traditional approaches to the translation could be substituted with the machine-aided method and underline the importance of further analyzing and sharing the experiences of employing the new technologies in the field of translation. Machines, especially neural networks, are considered in the context of the translation, education, and work with the mention of the outcomes. Different authors pointed out that using machine translation it is possible to improve the educational process, but at the same time, the activity of a human translator should have to be preserved.

It also presents a brief on competencies necessary for translators in the new world and an acknowledgment of post-editing in machine translation. Machine and automated translations are discussed and weighed and the most common programs used for translation such as DeepL, Google Translate, and Microsoft Bing Translator are discussed. The features of such programs are:

– the languages the programs support,
– various translation limits,
– an overview of other functions.

The analysis results that have been underlined are the further discussion regarding the effects of AI and computer programs in translation, the future research and generalization of the application of the technological advancement in translation, the importance of the MT as an add-on for learning particularly in the classroom setting, the shift of competencies of translators in the new technological environment, the necessity of post-editing of the MT and the distinction between MA and automatic translation [38].

Conclusions. Finally, this paper has expanded on the former existing research reviews on OSS advantages and disadvantage for practitioners by discussing the functional and commercial advantages and disadvantages done by the managers in the firms in the ESSS. Whereas the features like having the source code and being able to modify it has contributed in defining many technical advantages such as reliability, security, flexibility of use and performance. Within it, it was also established that such benefits were far superior to those of proprietary software. The business gains established in the research were also equal to the interviewees' gains equivalent to technical gains particularly the vendor locking avoidance, collaboration, and innovation gains. Nonetheless the user support from a community is very advantageous to OSS as whoever is employing the software is served by a proactive community of believers ready to assist with queries. Out of the identified firms, only one considered possible business advantage of adopting OSS as user support from the community.

The remaining companies indicated that support from the third party such as consultants, professional software houses were more appealing. Some technical disadvantages discovered in prior studies; for instance: different versions, installation issues, security issues, OSS is not as friendly and getting support and updating of OSS were not found to be serious limitations by the subjects, unlike proprietary software, OSS is less user-friendly and there was little evidence of companies having installation issues. Last but not the least, the business impacts discovered into the study reveal a similar picture as seen in the research findings of previous studies. Nonetheless, these disadvantages seemed to be a higher thorn in the flesh according to OSS than with their technical counterparts.

BIBLIOGRAPHY:

1. Setia P., Bayus B. L., Rajagopalan B. The takeoff of open-source software: a signaling perspective based on community activities. *MIS Quarterly.* 2020. Vol. 44.

2. Albeladi S. S. The role of open-source software to create digital libraries and standards assessment. *International Journal of Computer Science & Network Security.* 2021. Vol. 21. P. 241-248.

3. Fortunato L., Galassi M. The case for free and open-source software in research and scholarship. *Philosophical Transactions of the Royal Society A.* 2021. Vol. 379. Article number 20200079.

4. Boeing G., Higgs C., Liu S., Giles-Corti B., Sallis J. F., Cerin E., et al. Using open data and open-source software to develop spatial indicators of urban design and transport features for achieving healthy and sustainable cities. *The Lancet Global Health.* 2022. Vol. 10. P. e907-e918.

5. Lenarduzzi V., Taibi D., Tosi D., Lavazza L., Morasca S. Open-source software evaluation, selection, and adoption: a systematic literature review. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2020. P. 437-444.

6. Kholod I., Yanaki E., Fomichev D., Shalugin E., Novikova E., Filippov E., et al. Open-source federated learning frameworks for IoT: A comparative review and analysis. *Sensors*. 2020. Vol. 21. P. 167.

7. Bonati L., Polese M., D'Oro S., Basagni S., & Melodia T. Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead. *Computer Networks*. 2020. Vol. 182. Article number 107516.

8. Harris I. Package development. In *Beginning Salesforce DX: Versatile and Resilient Salesforce Application Development*. Springer, 2022. P. 457-529.

9. Green C., Amundson J., Garren L., Gartung P., Sexton-Kennedy E. SpackDev: Multi-package development with spack. *EPJ Web of Conferences*. 2020. Vol. 245. Article number 05035.

10. Cardoso M. J., Li W., Brown R., Ma N., Kerfoot E., Wang Y. et al. Monai: An open-source framework for deep learning in healthcare. *arXiv*. 2022. Vol. 2211. Article number 02701.

11. Winata A. P., Fadelina R., Basuki S. New normal and libraries services in Indonesia: A case study of university libraries. *Digital Libraries Perspectives.* 2021. Vol. 37. P. 77-84.

12. Fox E.A., da Silva Torres R. *Digital libraries technologies*. Springer Nature, 2022.

13. Cowell J. Managing a libraries service through a crisis. *Libraries Management.* 2021. Vol. 42. P. 250-255.

14. Kiron D., Spindel B. *Rebooting work for a digital era: how IBM reimagined talent and performance management*. MIT Press, 2020.

15. Boemer F., Kim S., Seifu G., de Souza F.D.M., Gopal V. Intel HEXL: accelerating homomorphic encryption with Intel AVX512-IFMA52. *arXiv*. 2021. Vol. 2103. Article number 16400.

16. Kato A., Kisangiri M., Kaijage S. A review development of digital libraries resources at university level," *Education Research International.* 2021. Vol. 2021. P. 8883483.

17. Chan T.T.W., Lam A. H. C., & Chiu D. K. From Facebook to Instagram: Exploring user engagement in academic libraries. *The Journal of Academic Librarianship.* 2020. Vol. 46. Article number 102229.

18. Lenarduzzi V., Tosi D., Lavazza L., Morasca S. Why do developers adopt open-source software? Past, present and future. In *IFIP International Conference on Open-source Systems*. Montreal: HAL Science, 2019. P. 104-115.

19. Fendt O., Jaeger, M.C. Open-source for open-source license compliance. In *Open-source Systems: 15th IFIP WG 2.13 International Conference, OSS 2019*. Montreal: HAL Science, 2019. P. 133-138.

20. Geyer-Blaumeiser L. *Ensuring open-source compliance using Eclipse Foundation technology*. EclipseCon Europe, 2022.

21. Azhakesan A., Paulisch F. Sharing at scale: an open-source-software-based license compliance ecosystem. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice*. ICSE-SEIP, 2020. P. 130-131

22. Yılmaz N., Kolukısa Tarhan A. Quality evaluation models or frameworks for open-source software: A systematic literature review. *Journal of Software: Evolution and Process.* 2022. Vol. 34. Article number e2458.

23. Spinellis D. How to select open-source components. *Computer*. 2019. Vol. 52. P. 103-106.

24. Urbančok B. D. *Blockchain open-source software comparison.* Masaryk University Faculty of Informatics (MUNI), 2019. URL: https://is.muni.cz/th/qr98z/thesis.pdf.

25. Billimoria K. N. *Linux Kernel Programming: A comprehensive guide to kernel internals, writing kernel modules, and kernel synchronization*. Packt Publishing Ltd, 2019.

26. Harutyunyan N., Bauer A., Riehle D. Industry requirements for FLOSS governance tools to facilitate the use of open-source software in commercial products. *Journal of Systems and Software.* 2019. Vol. 158. Article number 110390.

27. Karampatsis R.-M., Babii H., Robbes R., Sutton C., Janes A. Big code!= big vocabulary: Open-vocabulary models for source code. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. IEEE, 2020. P. 1073-1085.

28. Bonfield J.K., Marshall J., Danecek P., Li H., Ohan V., Whitwham A. et al. HTSlib: C libraries for reading/writing high-throughput sequencing data. *Gigascience.* 2021. Vol. 10. Article number giab007.

29. Campos C., Elvira R., Rodríguez J. J. G., Montiel J. M., Tardós J. D. Orb-slam3: An accurate open-source libraries for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics.* 2021. Vol. 37. P. 1874-1890.

30. Wang Y., Chen B., Huang K., Shi B., Xu C., Peng X., et al. An empirical study of usages, updates and risks of third-party libraries in java projects, In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020. P. 35-45.

31. Diesch R., Pfaff M., Krcmar H. A comprehensive model of information security factors for decision-makers. *Computers & Security.* 2020. Vol. 92. Article number 101747.

32. Bjork B.-C., Korkeamaki T. Adoption of the open access business model in scientific journal publishing: A cross-disciplinary study. *arXiv*. 2020. Vol. 2005. Article number 01008.

33. Tang J., Li B.-Y., Li K.W., Liu Z., & Huang J. Pricing and warranty decisions in a two-period closed-loop supply chain. *International Journal of Production Research.* 2020. Vol. 58. P. 1688-1704.

34. Grimaldi M., Greco M., Cricelli L. A framework of intellectual property protection strategies and open innovation. *Journal of Business Research.* 2021. Vol. 123. P. 156-164.

35. Enkel E., Bogers M., Chesbrough H. Exploring open innovation in the digital age: A maturity model and future research directions. *R&d Management.* 2020. Vol. 50.

36. Bouncken R.B., Kraus S., Roig-Tierno N. Knowledge-and innovation-based business models for future growth: Digitalized business models and portfolio considerations. *Review of Managerial Science.* 2021. Vol. 15. P. 1-14.

37. Trischler M.F.G., Li-Ying J. Digital business model innovation: toward construct clarity and future research directions. *Review of Managerial Science.* 2023. Vol. 17. P. 3-32.

38. Нестеров В., Костенко В., Курасова Н. Технологічні інновації у перекладі: вплив комп'ютерних програм та штучного інтелекту. *Вісник науки та освіти*. Том 20. № 2. С. 261-273.

REFERENCES:

1. Setia, P., Bayus, B. L., & Rajagopalan, B. (2020). The takeoff of open-source software: a signaling perspective based on community activities. *MIS Quarterly,* 44.

2. Albeladi, S. S. (2021). The role of open-source software to create digital libraries and standards assessment. *International Journal of Computer Science & Network Security,* 21, 241-248.

3. Fortunato, L., & Galassi, M. (2021). The case for free and open-source software in research and scholarship. *Philosophical Transactions of the Royal Society A,* 379, 20200079.

4. Boeing, G., Higgs, C., Liu, S., Giles-Corti, B., Sallis, J. F., Cerin, E., et al. (2022). Using open data and open-source software to develop spatial indicators of urban design and transport features for achieving healthy and sustainable cities. *The Lancet Global Health,* 10, e907-e918.

5. Lenarduzzi, V., Taibi, D., Tosi, D., Lavazza, L., & Morasca, S. (2020). Open-source software evaluation, selection, and adoption: a systematic literature review. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 437-444). IEEE.

6. Kholod, I., Yanaki, E., Fomichev, D., Shalugin, E., Novikova, E., Filippov, E., et al. (2020). Open-source federated learning frameworks for IoT: A comparative review and analysis. *Sensors,* 21, 167.

7. Bonati, L., Polese, M., D'Oro, S., Basagni, S., & Melodia, T. (2020). Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead. *Computer Networks,* 182, 107516.

8. Harris, I. (2022). Package development. In *Beginning Salesforce DX: Versatile and Resilient Salesforce Application Development* (pp. 457-529). Springer.

9. Green, C., Amundson, J., Garren, L., Gartung, P., & Sexton-Kennedy, E. (2020). SpackDev: Multi-package development with spack. *EPJ Web of Conferences*, 245, 05035.

10. Cardoso, M. J., Li, W., Brown, R., Ma, N., Kerfoot, E., Wang, Y. et al. (2022). Monai: An open-source framework for deep learning in healthcare. *arXiv*, 2211, 02701.

11. Winata, A. P., Fadelina, R., & Basuki, S. (2021). New normal and libraries services in Indonesia: a case study of university libraries. *Digital Libraries Perspectives,* 37, 77-84.

12. Fox, E.A., & da Silva Torres, R. (2022). *Digital libraries technologies*. Springer Nature.

13. Cowell, J. (2021). Managing a libraries service through a crisis. *Libraries Management,* 42, 250-255.

14. Kiron, D., & Spindel, B. (2020). *Rebooting work for a digital era: how IBM reimagined talent and performance management*. MIT Press.

15. Boemer, F., Kim, S., Seifu, G., de Souza, F.D.M., & Gopal, V. (2021). Intel HEXL: accelerating homomorphic encryption with Intel AVX512-IFMA52. *arXiv*, 2103, 16400.

16. Kato, A., Kisangiri, M., & Kaijage, S. (2021). A review development of digital libraries resources at university level," *Education Research International,* vol. 2021, p. 8883483.

17. Chan, T.T.W., Lam, A. H. C., & Chiu, D. K. (2020). From Facebook to Instagram: Exploring user engagement in academic libraries. *The Journal of Academic Librarianship,* 46, 102229.

18. Lenarduzzi, V., Tosi, D., Lavazza, L., & Morasca, S. (2019). Why do developers adopt open-source software? Past, present and future. In *IFIP International Conference on Open-source Systems* (pp. 104-115). Montreal: HAL Science.

19. Fendt, O., & Jaeger, M.C. (2019). Open-source for open-source license compliance. In *Open-source Systems: 15th IFIP WG 2.13 International Conference, OSS 2019* (pp. 133-138). Montreal: HAL Science

20. Geyer-Blaumeiser, L. (2022). *Ensuring open-source compliance using Eclipse Foundation technology*. EclipseCon Europe.

21. Azhakesan, A., & Paulisch, F. (2020). Sharing at scale: an open-source-software-based license compliance ecosystem. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice* (pp. 130-131). ICSE-SEIP.

22. Yılmaz, N., & Kolukısa Tarhan, A. (2022). Quality evaluation models or frameworks for open-source software: A systematic literature review. *Journal of Software: Evolution and Process,* 34, e2458.

23. Spinellis, D. (2019). How to select open-source components. *Computer,* 52, 103-106.

24. Urbančok, B. D. (2019). *Blockchain open-source software comparison.* Masaryk University Faculty of Informatics (MUNI). https://is.muni.cz/th/qr98z/thesis.pdf.

25. Billimoria, K. N. (2019). *Linux Kernel Programming: A comprehensive guide to kernel internals, writing kernel modules, and kernel synchronization*. Packt Publishing Ltd.

26. Harutyunyan, N., Bauer, A., & Riehle, D. (2019). Industry requirements for FLOSS governance tools to facilitate the use of open-source software in commercial products. *Journal of Systems and Software,* 158, 110390.

27. Karampatsis, R.-M., Babii, H., Robbes, R., Sutton, C., & Janes, A. (2020). Big code!= big vocabulary: Open-vocabulary models for source code. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, (pp. 1073-1085). IEEE.

28. Bonfield, J.K., Marshall, J., Danecek, P., Li, H., Ohan, V., Whitwham, A. et al. (2021). HTSlib: C libraries for reading/writing high-throughput sequencing data. *Gigascience,* 10, giab007.

29. Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., & Tardós, J. D. (2021). Orb-slam3: An accurate open-source libraries for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics,* 37, 1874-1890.

30. Wang, Y., Chen, B., Huang, K., Shi, B., Xu, C., Peng, X., et al. (2020). An empirical study of usages, updates and risks of third-party libraries in java projects, In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 35-45). IEEE.

31. Diesch, R., Pfaff, M., & Krcmar, H. (2020). A comprehensive model of information security factors for decision-makers. *Computers & Security,* 92, 101747.

32. Bjork, B.-C., & Korkeamaki, T. (2020). Adoption of the open access business model in scientific journal publishing: A cross-disciplinary study. *arXiv*, 2005, 01008.

33. Tang, J., Li, B.-Y., Li, K.W., Liu, Z., & Huang, J. (2020). Pricing and warranty decisions in a two-period closed-loop supply chain. *International Journal of Production Research,* 58, 1688-1704.

34. Grimaldi, M., Greco, M., & Cricelli, L. (2021). A framework of intellectual property protection strategies and open innovation. *Journal of Business Research,* 123, 156-164.

35. Enkel, E., Bogers, M., & Chesbrough, H. (2020). Exploring open innovation in the digital age: A maturity model and future research directions. *R&d Management,* 50.

36. Bouncken, R.B., Kraus, S., & Roig-Tierno, N. (2021). Knowledge-and innovation-based business models for future growth: Digitalized business models and portfolio considerations. *Review of Managerial Science,* 15, 1-14.

37. Trischler, M.F.G., & Li-Ying, J. (2023). Digital business model innovation: toward construct clarity and future research directions. *Review of Managerial Science,* 17, 3-32.

38. Nesterov, V., Kostenko, V., & Kurasova, N. (2024). Technological innovations in translation: the influence of computer programs and artificial intelligence. *Bulletin of Science and Education*, 2(20), 261-273.