Beyond Free: Hidden Costs as Major Contributors to Owning Open Source Software in Open Distance and E-Learning (ODEL) Departments of Universities in Embu and Kiambu Counties, Kenya

Duncan Kereu Kodhek[1]
John Wachira Kamau[2]

[1]duncankereukodhek@gmail.com
[2]jkamau@mku.ac.ke

[1]https://orcid.org/0009-0007-1005-6656
[2]https://orcid.org/0000-0002-5272-7733

[1,2]Mount Kenya University, Kenya

https://doi.org/10.51867/ajernet.6.2.11

ABSTRACT

The adoption of open-source software is growing, but at a slower rate than expected, despite its availability at no upfront cost. This is due to various direct, indirect, and hidden costs associated with ownership. Hidden costs, in particular, have a significant impact on the total cost of ownership, often outweighing direct and indirect costs. Understanding these cost factors is crucial for stakeholders to fully assess the financial implications. The Gartner's total cost of ownership theory was adopted for this study as it is a well-grounded theory in matters software cost estimation. The research design that was applied in this study is the explanatory sequential research design which involved the collection of both qualitative and quantitative data to better understand the concept of hidden costs in total cost of ownership of open source software. The population for the study was 316 participants in the open distance and learning departments (ODEL) in universities in Embu and Kiambu counties. The sampling technique used was the purposive sampling on the institutions that had customized and were using open source learning management system. The resultant sample size was 62 participants. The instrument used for data collection was a likert scale questionnaire. The data was analyzed using SPSS 24 and both descriptive and inferential statistics were applied on the collected data. The results obtained showed a relationship between direct cost factors and indirect cost factors to the total cost of ownership of open source software. Also, the hidden costs were identified as the greatest contributors to the total cost of ownership of open source software. Highlighting these hidden costs is essential for stakeholders to better track and manage them. This study contributes to the field by adapting Gartner's original model of total cost of ownership to open-source software. The recommendations for this study are that institutions should have a tracking mechanism for the cost factors of open source software so as to be able to cut down on costs. Also, government institutions are encouraged to track the costs of open source before choosing to adopt open source software.

Keywords: Adoption, Cost Factors, Learning Management Systems, Open-Source Software, Total Cost of Ownership

I. INTRODUCTION

Software refers to instructions instructing a computer or device on what to do. The software can be proprietary or open source. This classification is based on how one acquired the software and the availability of its source code (Asamoah, 2019). Proprietary software is software that one must pay to obtain, and its source code is not availed publicly. For one to have proprietary software, the buyers must part with cash to obtain them. The source code is kept private mainly because of the competitive nature of these software companies' businesses. Open-source software is a type of software that can be obtained for free or for a small fee (Asamoah, 2019). OSS's source code is open to the public. Open-source software is divided into two categories: desktop software and mobile application software. Desktop open-source software is simply software used on personal computers and laptops (Kamau & Sanders, 2017). On the other hand, open-source mobile application software is available on mobile devices. Institutions of higher learning in Kenya have put the question of Information Communication Technology (ICT) closer to the heart. Major institutions and organizations have got the aim of integrating ICT into their everyday life and making every aspect digitized. With this increase and quest for technological advancement, various firms and even the state will want to spend as little as possible on procuring ICT hardware and, most importantly, software (Kibuku et al., 2020) .In order to increase profits and minimize costs; several organizations thus choose to use open-source software instead of proprietary software.

Total cost of ownership of open source software (TCO) is defined as a tool that helps to account for all IT related costs. It therefore summarily concludes that TCO is a total of budgeted and unbudgeted costs that are

Licensed Under Creative Commons Attribution (CC BY-NC)

undergone throughout the lifecycle of IT equipment (Elsaid et al., 2021). In the recent past, only IT costs were added. Various organizations are adopting the TCO model to help in budgeting. TCO is said to be a tool that can be used to arrive at costs, for management, and for making purchasing decisions.

Institutions of higher learning in Kenya have put the question of technology closer to the heart. Major institutions and organizations have got the aim of integrating ICT into their everyday life and making every aspect digitized. With this increase and quest for technological advancement, various firms and even the state will want to spend as little as possible on procuring ICT hardware and, most importantly, software (Kibuku et al., 2020) .In order to increase profits and minimize costs; several organizations thus choose to use open-source software instead of proprietary software. Several institutions are thus adopting open source software so that they cut costs which are often attached to proprietary software (Kumar Jha, 2024).

Although open-source software (OSS) may help companies save a lot of money by doing away with license fees, there are typically hidden expenses to consider when adopting it. Expenses like this usually manifest in places like training, customization, and continuous support (Nagy et al., 2010). Staff may need specific training to operate and maintain the new systems efficiently after OSS implementation. The training programs offered by proprietary software providers are often extensive, whereas open source software (OSS) depends on community-driven resources, which may not be as organized or easily accessible. This calls for more funding to be allocated towards creating or acquiring suitable training materials and activities.

OSS may be customized to fit unique organizational requirements, it can be resource-intensive, but having the source code available allows for customizable solutions. Expenditures tend to rise since this method often necessitates employing competent developers to alter and integrate the program. In addition, future upgrades may not be compatible with heavily customized versions, which might need more modifications. Support is available from OSS communities via forums and documentation; but, enterprises that require quick answers may not be able to rely on this decentralized help. As a result, businesses may have to set aside money for internal support teams or hire outside vendors to fix technological problems as they crop up and keep systems running smoothly.

According to Böckelman (2017), conventional total cost of ownership (TCO) models for software often overlook a number of indirect and hidden costs that could have a major influence on a company's total spending. Training employees, tailoring the program to individual requirements, and providing continuous support and maintenance all contribute to these expenses. To help firms better understand the whole financial impact of software adoption and maintenance, Böckelman argues for TCO models that account for these ancillary expenses.

Another study by Han et al. (2021) focuses on hidden costs and says that they are often caused by maintenance, customisation, and integration issues, which are brought to light in the research that focuses on the OSS adoption landscape as a whole. When it comes to managing open source software (OSS) projects at major companies like Baidu, these hidden expenditures may have a considerable impact on the total cost of ownership, which isn't typically taken into consideration in initial cost evaluations (Han et al., 2021). According to the report, firms still have to spend a lot of money on technical knowledge, employee training, and software modification to fit their individual requirements, even though it may seem like a savings to not have to pay for licenses. The study points out that even though the cost of open source software are less, there are hidden cost that are attached to it which may make it to be expensive in the long run (Han et al., 2021).

1.1 Statement of the Problem

Undoubtedly, OSS adoption levels have risen over the years, and the challenge is not adoption anymore. There has arisen a new concern (Kereu, 2024). The model for TCO determination by Gartner is the closest to accuracy to measurement of TCO of software. It includes what is called cost factors and divides them into direct and indirect cost factors (Kereu, 2024). It however leaves out the hidden cost factors. Ilyas et al. (2021) finds out that there are hidden costs left out of the initial TCO models.

Enterprises risk making ill-informed decisions and incurring long-term financial constraints due to a lack of knowledge about the hidden costs associated with open-source software (OSS) adoption. Due to its lack of license costs, open source software (OSS) is sometimes positioned as an affordable substitute for proprietary software (Grzegorzewska et al., 2021). The hidden costs of its adoption, including those for customization, support, and training, are, however, often ignored or underappreciated by enterprises (Grzegorzewska et al., 2021). Poor financial planning and budgeting may lead to this mistake, which in turn can cause operational expenses to grow over time.

These hidden cost factors are the ones that this study seeks to highlight and show how they contribute to the TCO of OSS. Since such, the demonstration of the contribution of hidden cost factors in the OSS model is a major achievement, since it will provide institutions with a rigorous assessment framework with which to deal with the complexity of OSS adoption and prove to organizations how much hidden costs contribute to the costs they incur.

Licensed Under Creative Commons Attribution (CC BY-NC)

Kenyan and African IT firms and academic institutions would find this contribution particularly useful since it will inform their decisions on purchase in future.

1.2 Research Questions

The research problem above gives rise to the following questions which this study intends to solve:

i. What are the cost factors that influence the total cost of open-source software ownership in Kenyan Universities?
ii. What is the contribution of hidden costs to the model for total cost of open-source software ownership?

II. LITERATURE REVIEW

2.1 Theoretical Review

This work was grounded on Gartner's metric for TCO that was proposed in 1987. The metric is prominently used in IT and also in coming up with various methodological tools. The TCO model proposed by Gartner uses 2 major branches to categorize costs. The branches include direct costs and indirect costs (Heinrich et al., 2025). Direct costs can also be termed as budgeted costs are those costs that are anticipated by the organization when they are going to make a purchase they include; the fee, capital and the costs spent on labor by an organization's IT department so as to deliver various IT services to the enterprise and its users. The indirect costs are those costs not tracked in most organizations. Such costs include the time that may be spent by the user to get educated on how system works determination of the problem and downtime lost productivity. The Gartner's TCO model is the commonly used model and therefore it is treated as the standard (Heinrich et al., 2025).

The Gartner TCO model has been used in the school environment before in various case studies such as Kenya (Owoche, 2014) where the researcher came up with a TCO model to quantify the cost of ownership of the Maseno University ERP system. The TCO model developed suggests that the costs are divided into 5 cost elements which are: procurement costs, hardware and software costs, implementation costs, operations and maintenance costs and end user costs. The TCO model has also been applied to case studies of schools in USA districts such as Wisconsin and Virginia (Kuchina, 2024) The TCO model for Maseno University ERP system suggests that unbudgeted costs measure the effectiveness of the Information system giving the expected service to the user, if the developed system is well managed then users will not experience downtime and difficulties using the system (Owoche, 2014).

2.2 Empirical Review

2.2.1 Direct Cost Factors and Open Source Software

Fortunato and Galassi (2021) define the direct cost factors as the factors that influence the costs and can directly be pointed out by an organization. The direct costs are broadly divided into: search costs, acquisition costs and integration costs. Search costs are costs that are incurred by an organization as they make a decision to obtain software (Fortunato & Galassi, 2021). The costs entail the costs that are as a result of meetings held up to select a software, expert opinion that is sought after and demos and presentations that may require money to set up. Location and Operations, The expenses may mount up rapidly when meetings must take place off-site. If they don't have the necessary facilities on site, they could have to pay for leasing spaces. In addition, there can be transport and lodging costs if participants are travelling from distant places. Although these practical considerations are necessary to guarantee that all interested parties may take part in the decision-making process, proper budgeting is still necessary to keep costs in check.

An example of a search cost given is the money and resources that are spent on meetings (Fortunato & Galassi, 2021). Expert opinion charges apply in most of these instances, payments for specialized expertise and help throughout the software selection process are paid to consultants or industry experts via these fees. According to Fortunato and Galassi (2021) expert opinion is indeed needed, and this is a cost incurred during the search and thus worth to be included. Experts charge fees for their time and knowledge sharing, which may involve analyzing company needs, technical specifications, and possible software fit. Demonstrations and showcasing meant to understand the features of Open-source software is crucial and it is often carried out as suggested by Lanza-Cruz et al. (2024).

Acquisition costs in relation to the adoption are defined by Weder et al. (2022) of open-source software are the costs associated with the early steps of obtaining and setting up the software. These include pilot tests and installation fees. Before committing to a full-scale deployment, pilot testing is an essential stage in assessing the software's appropriateness and performance.

To ensure software testing does not affect live systems, organizations usually set up a restricted environment, also called a sandbox or pilot (Weder et al., 2022). An accurate evaluation of the software's performance and

Licensed Under Creative Commons Attribution (CC BY-NC)

compatibility may be achieved in this environment by simulating the organization's infrastructure and operations. To make sure the software is operational and ready for usage inside the organization, the installation price covers the initial expenses of obtaining and setting it up. Piloting of the a software is part of the acquisition phase and it adds to the costs of the system (Kereu, 2024)

Ilyas et al. (2023) defines it as the cost incurred as the user launches the system acquired so as to ensure that it works flexibly with the existing systems. Existing user migration is critical when integrating software systems, especially for software upgrades or platform transitions. Migrating user information from one system to another entails moving their profiles, preferences, data, and login credentials. Another aspect involved in integration costs is the development of application programming interfaces. Application programming interfaces, or PIs, are crucial because they allow various software systems to communicate and interact with one another.

Ofoeda et al. (2019) states that building application programming interfaces (APIs) allows for the smooth transfer of data, access to functionality, and compatibility when combining software components. Application programming interfaces (APIs) provide the standards, data formats, and authentication methods for integrating and enabling the seamless operation of different systems. Another crucial component in the integration cost is the development of plugins.

Plugins help the system software to add functionality to the software. Plugins are small, additional pieces of code that may be added to existing software to make it do more. When it comes to software updates and integration, creating plugins allows for personalization, extension, and connecting to other systems or services. To tailor a software platform to a user's needs, plugins might extend its functionality, connect to third-party APIs, or add new features (Negi et al., 2021).

2.2.2 Indirect Cost Factors and Open Source Software

Kula et al. (2021) describes indirect cost factors as factors that influence the costs however they cannot be directly pointed as to have been incurred out of use of software by an organization despite knowing that these costs exist. The costs that are indirect in an open source product are as discussed. Hiring costs is the amount that is spent on getting the relevant labor onboard so as to ensure that the integrated system functions as required (Chen et al., 2016). It is critical to think about adding new employees to the payroll when implementing a new system into a company. New systems often need specialized knowledge or abilities that present staff members may lack. The system's functioning and transition will run more smoothly with the help of qualified professionals brought in. New faces provide new ways of looking at things. When a team expands, it might provide fresh perspectives and ideas that the current members might have overlooked. More effective procedures and better results may result from this infusion of fresh ideas. The transition period after introducing a new system is also characterized by an increase in workload.

To avoid burnout and overwork among current workers, it is helpful to hire more people to divide up the responsibility. Instead of retraining current workers extensively, you may teach new hires immediately on the system—a great way to save time and money when implementing new systems (Chen et al., 2016). Recruiting new employees may help in managing change. Some workers may be resistant to using a new system because it disrupts their established processes. One way to encourage a culture of flexibility and openness to change is to staff positions with people who share this outlook.

The other indirect costs are training costs. These are costs that are incurred in the course of teaching the members available on how the newly acquired system functions. The instructions on how the new software functions are done by an expert who is often paid for services (Mugarza et al., 2020). The creation or acquisition of instructional resources like manuals, guidelines, or online course modules is not without its associated costs. The services of outside consultants and trainers, including their travel and lodging expenses in the case of on-site training, incur additional costs. The cost is further increased when the necessary infrastructure, such as classrooms, computers, and software licenses, is set up. The time and effort of internal trainers, as well as the expense of preparation and supplies, go into the sessions. While staff adjusts to the new system, there may be a short-term dip in output. To make sure workers get help using their new abilities, post-training support (whether internal or external) adds continuous expenditures (Mugarza et al., 2020).

Many factors affect how much it costs to host software. These include the kind of software (web-based, mobile app, desktop application), the amount of data transfer and users, the hosting provider, and any additional services needed (security, maintenance, support, etc.). The first thing you need to know is that "server space" is the amount of real or virtual storage space on a server that your software will be using (Snoussi, 2019). To make sure your software runs well and effectively, this area is crucial. Over a certain time frame, the term "bandwidth" refers to the amount of data sent and received by your system and its users. Important for figuring out how much data your hosting can handle, it includes both uploads and downloads. The term "data transfer" is synonymous with "bandwidth," which describes the amount of information sent and received by your software's server. Everything that people may see and

Licensed Under Creative Commons Attribution (CC BY-NC)

interact with on a website, including files, photos, and other forms of media, falls under this category of content. Another important consideration is load balancing, which involves spreading incoming network traffic across several servers (Negi et al., 2021). This method improves your software's speed and dependability by preventing any one server from being overburdened.

It is common practice to use third-party plugins or extensions to improve the functionality or add new features to open-source Learning Management Systems (LMS). While these plugins may greatly enhance your learning management system, they can also be rather expensive (Lanza-Cruz et al., 2024). Subscribers to third-party plugins may need to pay for updates, support, and new features on a regular basis. These subscription plans often include updates and support services, so you can be certain that your plugin will work with the most recent LMS versions and that any problems you have will be promptly resolved. If you want a third-party plugin to work with your learning management system (LMS) or meet your exact requirements, you may need to have it developed from the ground up. If you need a plugin built from the ground up or want to have an existing one modified, custom development services are the way to go. The level of customization needed determines the cost of these services, which might vary greatly (Turnbull et al., 2020).

Mugarza et al. (2020) centers their research on a zero-downtime paradigm, which means that any interruptions to the availability of the system are minimized or eliminated. They probably talk about compensation expenses in terms of how much money system downtime will cost. Compensations given to impacted parties, such workers or customers, for losses sustained during system outages might be a portion of the expenses. This may include not just the time wasted by workers as a result of inefficient work hours, but also the possible harm to the company's image and the faith of its customers. The impact of computer outages on employees is studied by Wang et al. (2019), who focus on the potential extra time employees may have to put in to make up for lost productivity. Various sorts of compensatory expenses may emerge. As an example, companies could have to pay workers more for overtime in order to compensate for the productivity they lost during downtime. Organizations may be obligated to provide impacted workers compensatory time off or other benefits in the event that downtime causes delays in workflow or deadlines.

2.2.3 Hidden Cost Factors and Open Source Software

Hidden cost factors are the ones that influence the costs however they cannot be directly be pointed out by an organization nor reflect in their books of account. France and Ghose (2019) define marketing of open-source software as an exercise that entails a number of tasks, while they are necessary for uptake and effective deployment, can pose a substantial hidden cost. Organizations may underestimate the total expenditure needed since these expenses are not usually considered explicitly when they choose to use open-source solutions. It is a frequent misunderstanding that open-source software is "free" due to the absence of upfront payments. The truth is that a lot of money has to go into introducing, implementing, and making the most of the software in a company (France & Ghose, 2019). A large portion of these funds will go towards raising public awareness among employees in the company. Adoption relies heavily on commercial and outreach initiatives. There will be design and production costs associated with creating marketing materials to showcase the LMS's capabilities and advantages. There are additional costs involved with community engagement via forums, social media, and conferences. These costs include staff time spent monitoring these platforms as well as charges for attending and sponsoring events. In order to set the product apart in a crowded software industry, the management of an organization at times hires marketing professionals who need to develop and convey a distinct value proposition (Alshare et al., 2019). This typically necessitates bringing in outside experts or using marketing firms.

Employee time is defined as the amount of time that an employee spends on software. Employee time, as it pertains to software, is the amount of time that people spend using certain software systems or apps for work-related tasks (Mowlawizadah, 2020). This encompasses many aspects of their software interaction that affect the organization's productivity and task management. When new or updated software is introduced, it usually takes some time for staff to get to know all the features and functions. Their workload may temporarily rise as they adjust to the new tools and procedures; becoming expert users may take more time due to this learning curve. Uptime for maintenance and system upgrades may be required when installing new features or updates, which may interrupt workflow and reduce productivity. Prior to complete integration into day-to-day operations, employees engaged in the implementation may have to set aside time to test and fix new features. In order to accomplish project goals or have the changeover go well, overtime could be required. Data transfer, software setup, and user assistance throughout the changeover are all examples of tasks that might need personnel working beyond regular business hours (Ngessa, 2019). Employees invest time into learning the system, incorporating it into their routines, and improving procedures to make the most of its features as they gain experience with it. In order to find the most effective and efficient processes, it is necessary to experiment with various configurations and settings during this adaptation period.

Licensed Under Creative Commons Attribution (CC BY-NC)

Dressen-Hammouda and Wigham (2022) defines retraining as the essentials in the software industry when workers want further explanation or clarification on concepts covered in earlier training. Organizations must provide more resources and time to cover this hidden cost, which may result from things like changes in software functionality. A part of retraining is having coworkers ask each other for help when they don't understand anything about the system. Though it's great for addressing problems and exchanging information, this informal consultation might affect productivity and workflow efficiency in the long run. When employees need a refresher on how to use the system, they might refer to workplace tutorials as a kind of retraining. These guides may be delivered in-person or made available online for use in the office. On the other hand, the organization has to pay for the resources used by these sessions, which include office space and internet bandwidth. To make sure workers fully grasp the software, some companies choose to have coaches meet with them one-on-one (Lehtola & Karttunen, 2022). There are obvious expenses connected with employing or outsourcing coaching services, but the benefits of having a coach or mentor provide personalized instruction to individual workers are worth the investment. Although retraining is necessary to keep employees proficient and productive, it comes at a hidden cost to businesses (Lehtola & Karttunen, 2022).

Attitude in this context refers to the motivation and willingness of the respondents to learn how the open-source software functions. One of the hidden costs that may affect the effectiveness of training programs and the overall learning curve is attitude, which plays a big part in how well new software systems are adopted inside an organization (Alshare et al., 2019). Negative attitudes towards change, training, or the new system may cause a lot of problems that make it harder for workers to get the skills they need. When new software is introduced, workers often respond with resistance to change since they are used to the status quo and are reluctant to try anything new. Because people may not be as motivated to fully engage with training materials or learning activities due to this resistance, the time it takes to become proficient could increase (Alshare et al., 2019). In addition, when employees have unfavorable attitudes towards training activities, such they find them boring or unimportant, their interest and participation might suffer. Employees' inability to retain and apply training material is a direct outcome of their attitude towards the sessions, which may be defined as indifference or disappointment. Employees may be less motivated to learn and grasp the new software system's capabilities if they have unfavorable opinions of it, such as reservations about its usability or relevance to their work functions (Negi et al., 2021). A lack of excitement and engagement during training sessions caused by this kind of attitude might make everyone take longer to get up to speed and put off enjoying the system's advantages.

III. METHODOLOGY

3.1 Research Design

The research design that was applied in this study is the explanatory sequential research design. This is a research technique that involves collection of quantitative data and then followed by conducting qualitative analysis to better understand the phenomena under study. The researcher collected quantitative data on a likert scale questionnaire and proceeded to analyze the collected data. The research design was used as it offered a deeper understanding of the results that were collected from the field.

3.2 Target Population, Sampling Techniques and Sample Size

3.2.1 Target Population

The population for the study was 316 participants in the Open distance and learning departments (ODEL) in universities in Embu and Kiambu counties. Kiambu County was chosen for study as it possesses the most universities pioneered adopting an open-source Learning management system (LMS). Embu County, on the other hand, was selected to bring about balance. The universities were selected through purposeful sampling. A preliminary study was conducted to identify the universities in the counties that used open-source Learning management systems. The Open Distance and Learning Departments (ODEL) department was picked because it is where the learning management system is often utilized. Thus, it has the population possessing characteristics desired by the user.

3.2.2 Sampling Techniques

A preliminary study first helped identify the number of universities that have customized OSS in Kiambu and Embu counties. Purposeful sampling was used to locate the universities in the two counties which have acquired and customized OSS LMS. Purposeful sampling is a technique in which the researcher chooses the subjects likely to possess the information he seeks. Purposeful sampling enables the researcher to avoid incurring costs that would otherwise be avoided. Finally, the researcher identified the number of respondents in each University through random sampling.

Licensed Under Creative Commons Attribution (CC BY-NC)

3.2.3 Sample Size Determination

One public university with the most participants and one private university with the most subjects in the population were purposefully selected for study. The purposefully selected institutions were subjected to 30% for the desired sample size. This gave rise to 62 participants as a sample.

Table 1

Sample Population for the Study

University	County	Population	Sampled participants
Kenyatta university	Kiambu	80	27
Mount Kenya University	Kiambu	50	15
Embu University	Embu	67	20
Total Sample			**62**

3.3 Research Instruments

The instruments that were used for data collection were Likert scale questionnaires. The questionnaire was a likert scale with strongly agree rated 1 and strongly disagree given a rate of 5. The questions were statements made so that the respondents could rate their level of agreement. The themes covered in the research questionnaire were broadly the aspect of direct cost factors, indirect cost factors and hidden cost factors of open-source software.

3.4 Data Collection

Data was collected by issuing questionnaires to the LMS users at the ODEL department. The research questionnaire was given filled and returned. The questionnaire was filled returned. The rationale behind the chosen format of collection was to ensure that the researcher had a better understanding of the cost factors.

3.5 Data Analysis and Presentation

The collected data was cleaned by removing the incomplete responses and leaving the complete ones. The responses given out by respondents were presented using tables, pie charts, and bar graphs for easier visibility. Further analysis was subjected to the questionnaires, whereby linear regression was applied to establish the relationship between variables.

IV. FINDINGS & DISCUSSION

4.1 Response Rate

Table 2 shows the response rate for questionnaires. All given out were filled and returned giving a 100% response rate. The response rate was good as the researcher ensured that the respondents understood their role properly and also promised to keep their responses confidential.

Table 2

Response Rate

Sampled	Responded	Response Rate
62	62	100%

4.2 Descriptive Statistics

The results are majorly presented through tables and graphs. They are as follows;

4.2.1 Descriptive Statistics on Direct Cost Factors

So as to find out whether direct cost factors the questions in the table below were posed to the respondents. The percentages show their responses based on every question. The respondents responded based on the Likert scale provided.

Licensed Under Creative Commons Attribution (CC BY-NC)

Table 3

The Responses on the Direct Cost Factors (N=62)

Question	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
(i) Search costs such as meetings to decide on LMS, expert opinion and demonstrations are applicable in open source LMS	67%	20%	13%	0%	0%
(ii) During acquisition installation fee, upgrades and piloting costs are incurred	55%	35%	10%	0%	0%
(iii) Integration activities like user migration, APIs, 3rd party plugins are necessary	80%	15%	5%	0%	0%

Table 3 shows the responses of the participants on the issue of the direct cost factors. The search costs such as decision meetings, expertise offered before purchase and demonstrations were identified and agreed upon by respondents. Majority of the respondents (87%) had an agreement with 13% being neutral. On the second cost element of acquisition, 90% of the respondents agreed that installation costs, upgrades and piloting takes up money upon encountering this element. The final cost element for direct costs where 95% of respondents agreed that integration is a direct cost element. Thus, search costs, acquisition costs and integration costs are direct cost factors.

4.2.2 Descriptive Statistics on Indirect Cost Factors

The responses summarized in the table that follows show the responses by respondents on the indirect cost factors. The questions were posed and the respondents were expected to respond with their level of agreement of the statement provided.

Table 4

Responses on Indirect Cost Factors (N=62)

Question	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
(i) Once LMS is in use, new staff, hosting fee and 3rd party monthly license fee is needed	87%	7%	6%	0%	0%
(ii) Compensation such as overtime costs are common for LMS	73%	26%	1%	0%	0%

Table 4 shows the responses on indirect cost factors. From the responses, majority of the respondents cited hiring costs, hosting fee and third party application licenses as indirect costs. 94% of the respondents were in agreement that these costs are part of indirect costs. Additionally, compensation costs such as overtime payments found a 99% support as indirect cost factors.

4.2.3 Descriptive Statistics on Hidden Cost Factors

The responses that are shown below are the ones given by participants on the hidden cost factors. The respondents were given statements and they required to indicate their level of agreement with the statements made. Table 5 below shows the responses given by the respondents.

Table 5

Responses on Hidden Cost Factors (N=62)

Question	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
(i) Marketing of LMS such as workshops, motivational talks are a norm before acquiring LMS	60%	27%	13%	0%	0%
(ii) Employee time is a cost that increases due to workload increase, customization and maintenance	47%	43%	3%	7%	0%
(iii) Retraining that happens through coaching, pre-recorded videos and expert consultation is a actually a hidden cost	50%	41%	9%	0%	0%
(iv) Attitude is a hidden cost as negative attitude leads to retraining	69%	25%	0%	6%	0%

Table 5 shows the responses that the respondents gave out concerning hidden cost factors. On marketing as costs 87% of the respondents were in agreement with the statement that marketing of the open source LMS, the

Licensed Under Creative Commons Attribution (CC BY-NC)

motivational presentations are necessary before the acquisition of the open source LMS. There were 13% of respondents who were neutral to the statement and did not give a clear response on the statement.

Employee time as a cost with workload increase, customization and maintenance as a cost got an agreement level of 90%. On the other hand, 3% of the respondents did not give a clear response but instead chose to remain neutral to that statement. Additionally, 7% of the respondents disagreed with the statement.

Retraining costs that happen through coaching of employees individually and prerecorded tutorials and expert consultations as a hidden cost received an agreement of 91%. The 9% of the respondents were neutral to the statement and did not give a clear response concerning it. Finally, Attitude as a hidden cost had an agreement of 94%. There were 6% respondents that disagreed with the statement on whether hidden costs contribute to the cost of an open source product.

4.3 Inferential Statistics

The results are presented below in form of tables with correlations between cost factors and the total cost of ownership of open source software. The first hypothesis was to determine whether the direct cost factors are significant to be in a relationship with the total cost of ownership of open source software.

Ho_1: There is a Significant Relationship between Direct Cost Factors and the Total Cost of Open-Source Software Ownership in Kenyan Universities

Table 6

Relationship between Direct Factors and Total Cost of Ownership

Correlations				
			Total Cost Ownership	**Direct Cost Factors**
Spearman's rho	Total Cost Ownership	Correlation Coefficient	1.000	.393**
		Sig. (2-tailed)	.	.002
		N	61	61
	Direct Cost Factors	Correlation Coefficient	.393**	1.000
		Sig. (2-tailed)	.002	.
		N	61	61

**. Correlation is significant at the 0.01 level (2-tailed).

Table 6 above shows the results obtained. There is a relationship between direct cost factors and total cost of ownership. The coefficient for the relationship is 0.393, which shows a moderate positive relationship between the two variables. This signifies that as the direct costs increase, the total cost of ownership also increases. The significance level is 0.002, showing that the relationship does not occur by chance or randomly. Therefore, there is a relationship between direct cost factors and total cost of ownership of open-source software. Hence H1 was accepted.

Ho_2: There is a Significant Relationship between Indirect Cost Factors and the Total Cost of Open-Source Software Ownership in Kenyan Universities

Indirect costs are known to be incurred by open-source software; however, they are only sometimes recorded in the account books. The research explored the indirect cost factors to ascertain whether the cost factors apply to open-source software. The results from the respondents are as explored below.

Table 7

The Relationship between Indirect Cost Factors and Total Cost of Ownership of Open-Source Software

Correlations				
			Total Cost Ownership	**Indirect Cost Factors**
Spearman's rho	Total Cost Owners Hip	Correlation Coefficient	1.000	.487**
		Sig. (2-tailed)	.	.000
		N	61	61
	Indirect Cost Factors	Correlation Coefficient	.487**	1.000
		Sig. (2-tailed)	.000	.
		N	61	61

**. Correlation is significant at the 0.01 level (2-tailed).

Licensed Under Creative Commons Attribution (CC BY-NC)

Table 7 above shows the relationship between indirect cost factors and total cost ownership. The Spearman's coefficient is 0.487, indicating a moderate positive relationship. As the indirect costs increase, the total cost of ownership also increases. The relationship is statistically significant; thus, it did not occur by chance or randomly. Therefore, there is a positive relationship between the total cost of ownership of open-source software and indirect cost factors. The relationship is statistically significant as seen, thus H2 was accepted.

Ho_3: There is a significant relationship between hidden cost factors and the total cost of open-source software ownership in Kenyan Universities.

The hidden cost factors are costs which are not accounted for and rarely seen as cost factors in an organization. The table below shows the relationship between the hidden cost factors and total cost of ownership of open-source software using the correlation coefficient.

Table 8

The Relationship between Hidden Cost Factors and Total Cost of Ownership of Open-Source Software

Correlations				
			Total Cost Ownership	**Hidden Cost Factors**
Spearman's rho	Total Cost Owners Hip	Correlation Coefficient	1.000	.594**
		Sig. (2-tailed)	.	.000
		N	61	61
	Hidden Cost Factors	Correlation Coefficient	.594**	1.000
		Sig. (2-tailed)	.000	.
		N	61	61

**. Correlation is significant at the 0.01 level (2-tailed).

Table 8 above shows that the correlation coefficient was 0.594. This value signifies a robust positive relationship between hidden cost factors and the total cost of ownership. This means that as hidden costs increase, so does the total cost of ownership. Therefore, the findings show a strong relationship between hidden cost factors and the total cost of ownership of open-source software. Thus, H3 was accepted. A comparison was done to determine which cost factors contribute most to the cost of open source software. Table 9 shows bellows the correlation coefficients of the cost factors

Table 9

Comparison of the Correlation Coefficients of the Cost Factors

Cost Factor	**Correlation Coefficient (ρ)**
Direct Cost Factors	0.393
Indirect Cost Factors	0.487
Hidden Cost Factors	0.594

Table 9 above shows the correlation coefficient for the cost factors. From the table it is evident that the hidden costs are the highest contributor to the total cost of ownership of open source software. The correlation coefficients demonstrate the intensity of the association between each cost element and the total cost of ownership (TCO). Therefore, the hidden cost factors are the highest ranking in the cost factors.

4.4 Discussion

4.4.1 Answering the First Research Question

The analysis showed the relationship between direct cost factors and the total cost of ownership of open-source software. The direct cost factors, as found out by the study, include the search costs, which are costs incurred when deciding on the type of software to select. The search costs include selection meetings, demonstration, and presentation costs. Another direct cost factor is integration costs that entail user migration, API development, and plugin costs. The other direct cost is acquisition costs, including installation and pilot test fees. The finding is in line with a study that was carried out by Maha, whose study shows that even though several people perceive open-source costs to be zero, there are costs associated with acquisition, integration, and support, which are direct costs. The study also points out that the direct cost factors contribute significantly to the total cost of ownership of open-source software (Shrestha & Sheikh, 2021).

Licensed Under Creative Commons Attribution (CC BY-NC)

Secondly, the study showed that the indirect cost factors enormously contributed to the total cost of ownership of open-source software. The indirect cost factors, as found out by this study, included hiring new staff, hosting costs for the software, and costs incurred on third-party software licenses. The findings agree with studies by Owoche, which found that indirect cost factors such as hiring new staff and hosting costs contributed significantly to the total cost of ownership of the software (Owoche, 2014). Another study by Ngessa found that indirect costs contributed more than direct costs (Ngessa, 2019).

4.4.2 Answering the Second Research Question

The second research question sought to determine what cost factor contributes the most to the total cost of ownership of open source software. The correlation coefficients between the cost factors and the total cost of ownership of open source software found that the hidden cost factors had a higher coefficient.

The hidden costs comprise the open-source learning management system comprising awareness costs and professional opinion fees. *Employee time* is the second attribute comprising workload increase allowances, downtime, and overtime fees (Kereu, 2024). Another element of the hidden costs is retraining, which consists of consultation fees, tutorial costs, and one-on-one coaching fees. The final element is attitude, which includes change resistance and negative attitude.

V. CONCLUSION & RECOMMENDATIONS

5.1 Conclusion

In conclusion, this study highlighted the significant role of hidden costs in the total cost of ownership of open-source software. By applying Gartner's Total Cost of Ownership theory, the research revealed that while direct and indirect costs are important, hidden costs are the primary contributors to overall ownership expenses. This finding underscores the need for organizations and stakeholders to recognize and quantify these hidden costs, enabling more informed decision-making when adopting open-source software. Understanding the full financial impact of these costs will help organizations better manage and track their investments in open-source solutions.

5.2 Recommendations

Institutions of higher learning and other organizations should weigh the direct, indirect, and hidden costs of open-source software carefully. They need to look at the short-term costs and benefits of using open-source technologies by doing a cost-benefit analysis. Budgeting for acquisition, integration, and maintenance should be a primary emphasis of medium-term strategy. It is also important to have cost-tracking measures in place to monitor continuing spending. A cost-conscious culture that encourages efficient usage and contribution to the open-source community should be fostered at institutions in the long run. The government should also make it a requirement that ministries adopting open-source software disclose all expenses, including indirect and hidden ones, in order to maximize efficiency and minimize wasteful expenditure.

REFERENCES

Alshare, K. A., Alomari, M. K., Lane, P. L., & Freeze, R. D. (2019). Development and determinants of end-user intention: Usage of expert systems. *Journal of Systems and Information Technology, 21*(2), 166–185. https://doi.org/10.1108/jsit-08-2018-0108

Asamoah, M. K. (2019). Reflections and refractions on Sakai/Moodle learning management system in developing countries: A case of Ghanaian universities' demand and supply perspective analyses. *African Journal of Science, Technology, Innovation and Development, 12*(2), 243–259. https://doi.org/10.1080/20421338.2019.1634318

Böckelman, C. (2017). *Cost analysis of cloud based converged infrastructure for a small sized enterprise* (Master's thesis, Aalto University). https://aaltodoc.aalto.fi/items/446eb666-f68b-41f4-a2a2-7ce3400daf1a

Chen, X., Zhou, Y., Probert, D., & Su, J. (2016). Managing knowledge sharing in distributed innovation from the perspective of developers: Empirical study of open source software projects in China. *Technology Analysis & Strategic Management, 29*(1), 1–22. https://doi.org/10.1080/09537325.2016.1194387

Dressen-Hammouda, D., & Wigham, C. R. (2022). Evaluating multimodal literacy: Academic and professional interactions around student-produced instructional video tutorials. *Education in the Knowledge Society, 105*(2), Article 102727. https://doi.org/10.1016/j.system.2022.102727

Licensed Under Creative Commons Attribution (CC BY-NC)

Elsaid, M. E., Abbas, H. M., & Meinel, C. (2021). Virtual machines pre-copy live migration cost modeling and prediction: A survey. *Distributed and Parallel Databases, 40*(2), 1–35. https://doi.org/10.1007/s10619-021-07387-2

Fortunato, L., & Galassi, M. (2021). The case for free and open source software in research and scholarship. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 379*(2197), 20200079. https://doi.org/10.1098/rsta.2020.0079

France, S. L., & Ghose, S. (2019). Marketing analytics: Methods, practice, implementation, and links to other fields. *Expert Systems with Applications, 119*, 456–475. https://doi.org/10.1016/j.eswa.2018.11.002

Grzegorzewska, P., Katz, A., Muto, S., Pätsch, S., & Schubert, T. (2021). *The impact of Open Source Software and Hardware on technological independence, competitiveness and innovation in the EU economy* (pp. 320–335). European Commission. https://www.ospi.es/export/sites/ospi/documents/documentos/CNECT_OpenSourceStudy_EN_28_6_2021_LMBhSihnCeC7JEDsHXkK1JlZ0_79021_compressed.pdf

Han, J., Deng, S., Lo, D., Zhi, C., Yin, J., & Xia, X. (2021). An empirical study of the landscape of open source projects in Baidu, Alibaba, and Tencent. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. https://arxiv.org/abs/2103.01590

Heinrich, A., Putz, F., Krollmann, S., Loss, B., Ahmed, W., & Hollick, M. (2025). GWEn -- An open-source wireless physical-layer evaluation platform. *arXiv Preprint*. https://arxiv.org/abs/2501.13144

Ilyas, M., Khan, S. U., Khan, H. U., & Rashid, N. (2023). Software integration model: An assessment tool for global software development vendors. *Journal of Software: Evolution and Process, 36*(4), e2540. https://doi.org/10.1002/smr.2540

Kamau, J., & Sanders, I. (2017). Adoption of free desktop open source software in developing countries in Africa: A case of Kenyan university students. *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, 240–255. https://core.ac.uk/download/pdf/154915166.pdf

Kereu, D. (2024). A model for total cost determination in open-source software ownership: Case of Kenyan universities' learning management system. *Computer and Information Science, 17*(1), 1–18. https://ideas.repec.org/a/ibn/cisjnl/v17y2024i1p18.html

Kibuku, R., Ochieng, D., & Wausi, A. (2020). E-learning challenges faced by universities in Kenya: A literature review. *The Electronic Journal of E-Learning, 18*(2), 225–249. https://doi.org/10.34190/EJEL.20.18.2.004

Kuchina, M. (2024). *Learning management system integration plan to current project-based customer training process* (Bachelor's thesis, Metropolia University of Applied Sciences).

Kula, E., Greuter, E., Van Deursen, A., & Georgios, G. (2021). Factors affecting on-time delivery in large-scale agile software development. *IEEE Transactions on Software Engineering, 48*(9), 1–10. https://doi.org/10.1109/tse.2021.3101192

Kumar Jha, S. (2024). Adoption and economic impact of open-source software in digital libraries. *African Journal of Biomedical Research, 27*(5), 719–725. https://doi.org/10.53555/ajbr.v27i5s.6755

Lanza-Cruz, I., Montoliu Colás, R., Martínez-Martínez, A., & Quintana, I. R. (2024). Advancing virtual learning: A review of Moodle for the optimization of online education. *INTED Proceedings, 24*(2), 4056–4064. https://doi.org/10.21125/inted.2024.1040

Lehtola, S., & Karttunen, A. J. (2022). Free and open source software for computational chemistry education. *WIREs Computational Molecular Science, 22*(3), e1610. https://doi.org/10.1002/wcms.1610

Mugarza, I., Parra, J., & Jacob, E. (2020). Cetratus: A framework for zero downtime secure software updates in safety-critical systems. *Software: Practice and Experience, 50*(8), 1399–1424. https://doi.org/10.1002/spe.2820

Nagy, D., Yassin, A. M., & Bhattacherjee, A. (2010). Organizational adoption of open source software. *Communications of the ACM, 53*(3), 148–151. https://doi.org/10.1145/1666420.1666457

Negi, S., Rauthan, M. M. S., Vaisla, K. S., & Panwar, N. (2021). CMODLB: An efficient load balancing approach in cloud computing environment. *The Journal of Supercomputing, 77*(8), 8787–8839. https://doi.org/10.1007/s11227-020-03601-7

Ngessa, V. (2019). A study of basic costs for tracking total cost of ownership of information and communication technology in an academic institution: The case of the Institute of Accountancy Arusha. *International Journal of Advanced Research, 7*(11), 228–234. https://doi.org/10.21474/ijar01/9997

Ofoeda, J., Boateng, R., & Effah, J. (2019). Application programming interface (API) research. *International Journal of Enterprise Information Systems, 15*(3), 76–95. https://doi.org/10.4018/ijeis.2019070105

Owoche, P. (2014, March 28). *Model for total cost of ownership evaluation of enterprise resource planning: Case of Maseno University* [Manuscript]. Academia.edu. https://www.academia.edu/6585477

Licensed Under Creative Commons Attribution (CC BY-NC)

Shrestha, L., & Sheikh, N. J. (2021). Multiperspective assessment of enterprise data storage systems: The use of expert judgment quantification and constant sum pairwise comparison in finding criteria weights. *Open Journal of Business and Management, 9*(2), 955–980. https://doi.org/10.4236/ojbm.2021.92051

Snoussi, T. (2019). Learning management system in education: Opportunities and challenges. *International Journal of Innovative Technology and Exploring Engineering*, *8*(12S), 664–667. https://doi.org/10.35940/ijitee.l1161.10812s19

Turnbull, D., Chugh, R., & Luck, J. (2020). Learning management systems: An overview. In *Encyclopedia of Education and Information Technologies* (pp. 1052–1058). Springer. https://doi.org/10.1007/978-3-030-10576-1_248

van der Meij, H., Rensink, I., & van der Meij, J. (2018). Effects of practice with videos for software training. *Computers in Human Behavior, 89*, 439–445. https://doi.org/10.1016/j.chb.2017.11.029

Wang, C., Hsu, H.-C. K., Bonem, E. M., Moss, J. D., Yu, S., Nelson, D. B., & Levesque-Bristol, C. (2019). Need satisfaction and need dissatisfaction: A comparative study of online and face-to-face learning contexts. *Computers in Human Behavior, 95*, 114–125. https://doi.org/10.1016/j.chb.2019.01.034

Weder, F., Yarnold, J., Mertl, S., Hübner, R., Elmenreich, W., & Sposato, R. (2022). Social learning of sustainability in a pandemic—Changes to sustainability understandings, attitudes, and behaviors during the global pandemic in a higher education setting. *Sustainability, 14*(6), 3416. https://doi.org/10.3390/su14063416

Licensed Under Creative Commons Attribution (CC BY-NC)