Computer and Information Science; Vol. 17, No. 1; 2024
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

# A Model for Total Cost Determination in Open-Source Software Ownership: Case of Kenyan Universities' Learning Management System

Duncan Kereu Kodhek[1], John Wachira Kamau[1], & Faith Mueni Musyoka[2]

[1] School of Computing and Information Technology, Mount Kenya University, Thika

[2] Department of computing and Information Technology, University of Embu, Embu

Correspondence: Duncan Kereu Kodhek, School of Computing and Information Technology, Mount Kenya University, Thika, Kenya. E-mail: duncankereukodhek@gmail.com

Received: January 25, 2024 Accepted: March 1, 2024 Online Published: March 8, 2024

doi:10.5539/cis.v17n1p18 URL: https://doi.org/10.5539/cis.v17n1p18

**Abstract**

The adoption of open-source products is slowly increasing; the increase, however, is slower than expected, considering that most open-source products are freely available. Researchers and scholars have attributed the adoption levels to, among other things, a lack of know-how of the total cost of ownership of the open-source software. Thus, it is crucial for the cost of owning the software to be developed. While an ongoing endeavor to develop a model to determine the total cost of ownership of open-source software, the models have proved to be less accurate and do not capture essential elements. Moreover, there has been a rising call for organizations to adopt open-source software to lower the software costs incurred on proprietary software. If the cost of owning open-source software were known, it would be beneficial as several organizations and institutions could adopt it readily. The data was collected from Universities in Kiambu and Embu Counties. Linear regression analysis was done to help develop the model, and a mathematical model was developed. The proposed model was: total cost of open-source software ownership = direct + +indirect + hidden costs. To validate the model, it was subjected to expert validation. The model will be an outstanding contribution to information technology as it will make it possible to come up with the total cost of owning open-source software.

**Keywords:** adoption, open-source software, total cost ownership

## 1. Introduction

The definition of open-source software is given as software available for one to use either freely or at a small fee. However, the source code is freely available for modification and alteration (Wiley, 2014). A successful demonstration of open-source software is the now widely used Moodle Learning management system. There are other older open-source products, such as the famous operating system -Linux.

The open-source products have found themselves in the hands of institutions such as Universities in Learning management systems such as Sakai and Moodle to save an extra coin. Open-source software is believed to carry several advantages compared to its counterpart proprietary software, whose primary drive is profits (Mthethwa-Kunene & Maphosa, 2020). The advantages of open-source software include saving on costs as it is acquired freely. The other advantage is enormous community support since several people have the source code and thus propose changes and improvements to the original product. Thus, due to the above advantages, institutions, companies, and governments are calling for switching to open-source software to slash costs (Asamoah, 2019).

Despite the adoption to cut costs, organizations are skeptical about the actual cost of open-source software. Several organizations have halted the quest to adopt the open-source product. The reason for this is the lack of a way to determine the total cost of this software (Bista, 2023). The Gartner group first proposed the total cost of software ownership concept upon finding they needed help with some software costs. The group thus devised a model that would use cost accounting to obtain the total cost of software ownership. The Gartner Group defines Total cost of ownership as all the costs that involve information technology and anything encompassing information technology (Gollapudi et al., 2018). Therefore, the total cost of software ownership is the entire cost, from the planning of acquisition to the final stage of the software, which is retirement.

### *1.1 Research Problem*

According to research on Open-source software adoption in institutions, as of 2012, 60% of organizations in Kenya had adopted Open-source software (Gichira et al., 2012). Undoubtedly, Open-source software adoption levels have risen over the years, and the challenge is not adoption anymore. There has arisen a new concern. The total cost of ownership of open-source software is unknown by organizations (Kamau, 2017). This is the problem that this study seeks to address. The study intends to develop a Model to calculate the total cost of owning open-source software. There are available models of the total cost of ownership; however, they cannot come up with the total cost of ownership of open-source software. The generic metrics need to be improved, as cited by (Bensberg & Dewanto, 2003), where it is argued that the total cost of ownership uses cost accounting instead of capital budgeting.

On the other hand, Bailey & Heidt, (2003) counters that the available models are futuristic and thus not helpful. Whereas Owoche (2017) argues that the original Gartner model was not specific to a particular scenario and thus ineffective. Böckelman, (2017) says that the generic total cost of ownership models fails to factor in risk. Finally, Ilin et al. (2021) discovered hidden costs left out of the initial TCO models. The aim of open-source software is cost reduction and an increase in cost is the least expected scenario. How can we then determine an approximate price before committing to adoption?

There is a need to determine the total cost of ownership of open-source software in Kenya and Africa (Kamau, 2017). The existing models measure other aspects that are not open-source software. For instance, the research carried out by Taibi, Lavazza, and Morasca (2007) highlighted that most organizations did not want to adopt open-source software mainly because there was no well-structured evaluation technique. The organizations needed a proper open-source software model to evaluate costs. Bassi (2010) argues that the total cost of ownership available is not specific to open-source software, and thus, there is a need for a model to determine open-source software. Also, because of the unique nature of the open-source software, it has to use a different procurement style, which demands a new model to aid in making procurement decisions (Golden, 2005).

### *1.2 Justification*

The proposed model gives institutions a well-structured evaluation tool to enable them to adopt open-source software. It significantly contributes to Kenyan and African continent Information Technology organizations and institutions of higher learning as there is a lack of contextualized models for open-source software. The model addresses the issue of hidden costs left out by the generic total cost of ownership models. It factors hidden costs and is contextualized to an open-source learning management system. Kenyan and African information technology firms and academic institutions find this contribution particularly useful since it tackles the unique contextual difficulties they encounter.

### *1.3 Purpose of the Study*

This study aimed to develop a model for total cost determination in open-source software ownership.

## 2. Literature Review

This part of the review explores the literature that has been done regarding cost of software. It explores the techniques, the methods and models of total cost of ownership of open-source software.

### *2.1 Techniques to Calculate Cost of Software*

Techniques are generally acceptable ways of doing something. The techniques to calculate the software cost are proven to work when quantifying the cost. The following table below summarizes the techniques used to quantify the cost of owning software, their definition, and the limitations of the techniques, as pointed out by various scholars.

#### 2.2.1 Expert Judgment

As expounded by Shrestha and Sheikh (2021), the technique involves taking the knowledge on the cost of software from experts. The technique involves first identifying the project one wants to undertake, the second step is pinpointing experts skilled in software development and estimation, and the third step is giving out their views where the expert view of every member is taken into consideration. Finally, the judgments were analyzed, and an average was carried out through such methods as weighted averaging. The technique however has drawbacks.

The technique's most significant limitation is that it is prone to bias. As DeMarco and Lister (2013) pointed out, the biggest challenge is bias and being subjective on the side of the experts. The judgment of the experts, as

pointed out by the authors, is subjective to their views on the subject. Another disadvantage of the technique highlighted by the work is inconsistency among the experts, as different experts may explain a similar situation based on their view on that particular subject.

#### 2.1.2 Software Estimation by Analogy

It is a software estimation technique that is based on comparisons. A project is taken and checked based on its resemblance to a past project (Phannachitta, 2020). The procedure followed in this kind of estimation is to identify a similarity between the project that has been identified and the one you are about to carry out; the second step is to collect data from the historical project pointed out, and the third step entails analysis of the metrics used to identify trends that can be related to the current project. Finally, the differences are adjusted, and the model is applied to the current project. The technique of assessing a project with a similar one in the recent past has its disadvantages.

As Katoh and Standley (2013) pointed out, the disadvantage of estimation by analogy is first that the technique is built upon assumptions. The technique first assumes that the projects with certain similarities will likely cost the same amount of cash. This assumption often needs to be revised, as pointed out by the article. Another disadvantage is the difficulty of finding data that aligns with a project one wishes to undertake.

#### 2.1.3 Parkinson's Law Software Estimation

It is a general law that states that regardless of the effort that is applied, work or a project expands to be done within the time frame that it was scheduled to be done. The shortcoming of the technique is its assumption that effort is always well-spent.

#### 2.1.4 Pricing to Win Technique of Software Estimation

This is a technique used for software estimation whereby the cost is taken to be what a customer offers (Thota et al., 2020). For instance, if a customer offered 200 dollars for a software project, the cost would be 200. However, the technique of molding software according to a client's budget has disadvantages. As pointed out by Thota et al. (2020), a limitation of this technique is the likelihood of compromising user requirements and features for the money the customer gives.

#### 2.1.5 Bottom-up and Top-down Estimation Techniques

The bottom-up software estimation technique takes the cost of every module, adding it as software development goes on. The bottom-up starts with small patches joining them to make up the more significant project (Jørgensen, 2004). On the other hand, the top-down software estimation technique contemplates a software project as one massive project. Then, it cuts it down into smaller, understandable components or modules. The techniques have drawbacks that render them unsuitable. As Tayal et al. (2019) demonstrated, the methods often ignore the crucial step of documentation. Also, the techniques are time-consuming and resource-consuming since the breakdown of a project into its modules is a laborious task and, thus, an expensive exercise in the long run. The techniques are also known as traditional ways of software estimation. Over time, better and more sophisticated methods emerged. Unlike a technique, the model may incorporate several techniques and methods to improve cost estimation.

### *2.2 Software Ownership Models*

A model describes how structures function and how computation is arrived at. It can also be described as a representation of a complex scenario in a more straightforward way that the audience can easily understand.

The models are classified as either mathematical or pictorial. Mathematical models are in the form of equations that can be used to give out a figure. On the other hand, pictorial models are diagrammatical representations that depict the flow of functions and structures. Below is a table summarizing software ownership models, their descriptions, and the corresponding limitations.

#### 2.2.1 COCOMO

It is an acronym for constructive cost model. It was developed by Barry Boehm in 1981 after collection of data from several software development projects. Since its development in 1981, the model has undergone various improvements and updates. The model is given as an equation that factors the software project's time, size, and effort.

The equations for the COCOMO are listed below:

Basic COCOMO model

$$E=a\times(KLOC)b \quad \textbf{(1)}$$

With E being effort calculated in person-months, a and b being constants obtained from historical projects, and KLOC (kilobytes in lines of code). The above equation was the basic COCOMO equation proposed in 1981.

The basic model was criticized by Ilham Cahya Suherman et al. (2020) because the model should have taken into consideration other factors, such as environmental factors. These weaknesses gave birth to the next version of COCOMO.

*Intermediate COCOMO model*

This version was conceptualized to cover the defects in the basic COCOMO. The intermediate COCOMO incorporated 15 cost drivers into its initial equation, thus improving it. The equation for the intermediate model is given as:

$$E=a\times(Size)b\times\prod i=1nEmi. \quad (2)$$

The equation considered personnel attributes, hardware factors, and environmental factors.

n is the number of cost drivers presented.

A and b are the constants obtained from the historical projects.

E is effort in person-months, and *Emi is* the cost driver.

*The detailed model*

This is an improvement of the intermediate model; it retains the equation above while it introduces an assessment of the cost drivers in the six stages of software development. The equation for the model is:

$$E=a\times(Size)b\times EAF. \quad (3)$$

Where E is the effort in person-months, a and b are constants, EAF is a product of various cost drivers.

2.2.2 Estimacs Model

Robin Howard put forth the model. It is applicable mainly in finding the approximate cost of safety-critical software. The model relies on function points to give the software cost. The equation applied to come up with an approximate cost is private as it is proprietary software (Kahuti, 2020). The model involves function points, which are 25 input functionalities rather than lines of code. The 25 questions are pre-defined and awarded weights used to arrive at a weighted score. The score is subjected to an internal classified algorithm, and an estimate of the software project is given. The model however had some disadvantages that made it unsuitable. As illustrated by Baghel et al. (2020), the most significant limitation of the model is the need for more transparency on how data is manipulated and is thus prone to bias.

2.2.3 PUTNAM Model

It is also known as the software lifecycle model (SLIM). It predicts software cost in person-months by measuring the estimated lines of code in conjunction with several other factors (Sharma et al., 2011). The equation for the model is given as:

$$\text{Effort (PM)} = B * L \wedge Ck * Td \wedge D0 \quad (4)$$

Where effort is given in PM (person-months), B is the factor of scale, L is the software size in lines of code, and Ck is the development team's experience. On the other hand, Td is development time, and D0 is the effort multiplier.

Although the model is presented as a classic, it has some faults which make it less suitable. For instance, Sharma et al. (2011) criticize the model as using historical data that needs to reflect the changing technology in the industry.

2.2.4 SEER-SEM Model

The model was put up by Golath in 1983. The model works on simulation and emulates how the final software will appear. The model has inputs and outputs (Saleem et al., 2019). The inputs are the size of the project, the personnel needed for the development, the environmental characteristics, the constraints of the project, and the software project's complexity level. The inputs are entered by a user or through analysis of historical data or expert opinion.

The outputs are approximated in cost, effort, schedule, maintenance, and reliability.

The basic equation is given as:

$$\text{Effort (E)} = \text{Size (S)} * \text{Productivity Factor (PF)} \quad (5)$$

The major limitation has been said by researchers to be that the model has complex internal calculations, which makes it undesirable to several project managers as it is difficult for them to comprehend how it functions (Wiley, 2014).

*2.3 Total Cost of Software Ownership Models*

The term total cost of ownership was coined by the Gartner Group in 1987 when they discovered that costs were not reflected in their books of account. The cost of owning software covers every aspect, from purchase to retiring the software. Below are the total cost of ownership models.

2.3.1 Gartner Total Cost Ownership Model

This was the very first model developed by the Gartner group so that they could account for the costs of the computing machinery that was available on their premises. The model goes beyond the purchase price of hardware or software equipment to include its service fee and other costs incurred. The model has postulated direct and indirect costs (Kumar & Behera, 2020). The indirect costs could be accounted for and reflected in books of account. Indirect costs, on the other hand, are derived costs that may not be reflected in the books of accounts.

$$\text{TCO= direct costs+ indirect costs} \quad (6)$$

TCO- Total cost of ownership.

The model comprises five phases: acquisition, deployment, management, end user, and disposal costs. The acquisition phase entailed the upfront costs of hardware and software, training and license fees, and installation costs. The second phase included integration costs, customization fees, testing, and data migration fees. The third phase entails management costs for extra staff, maintenance, and security. The fourth phase was end-user costs, which included downtime, lost time, and a learning curve. The final phase had disposal costs, which included data removal, de-installation, and recycling. The Gartner model is criticized as having limitations by various scholars.

Owoche (2017) argues that the Gartner group model was not tailored to any particular product. However, it was a generic model, and thus, there are more suitable models for estimating software costs. Also, the writer critiques the model as a cost accounting tool that comes in handy after the damage has been done; thus, it cannot be used for prediction or planning.

2.3.2 Total Cost of Ownership Model for a Server Platform

The model is an equation that is mathematical and is applied to come up with the cost of owning a cloud server platform (Capuano, 2014). The equation for the model is given as:

$$\text{TCO} = \text{Chw} + \text{Csw} + \text{Csvc} + \text{Coth} \quad (7)$$

Where: TCO is the total cost of ownership, **Chw-** is the cost of hardware, **Csw** is the cost of software, **Csvc** is the cost of service, and **Coth** is any other costs. The model has faults and has been critiqued by Chi et al. (2021). The scholar argues that the other costs left open in the equation (**Coth**) are often ignored when coming up with costs since they have yet to be specified.

2.3.3 Total Cost of Ownership of ERP System

The model for an enterprise resource planning system was developed by Patrick Owoche in 2017. The model. It was developed after careful examination of the system at Maseno University. The model comprises procurement, hardware and software acquisition, implementation, maintenance, and end-user costs.

The study's limitations, as pointed out by Akbar (2019), the model cannot be generalized as it was done as a case study in a single institution. Also, the article needs to fix the model as a data-collecting tool rather than a software estimation model.

2.3.4 Cloud Computing Model

The cloud computing model was developed by McKinsey & Company. The model aims to calculate the approximate cost (Gadatsch, 2023). The equation that was created for the total cost of ownership of the cloud is given as:

$$\text{TCO= Initial Costs + Recurring Costs – Savings} \quad (9)$$

TCO is the total cost of owning the cloud. It is given by adding the initial and recurring expenses and subtracting the savings. It is a model that involves additional costs recurring costs and removes the savings incurred in a system.

*2.4 Total Cost of Open-Source Software Ownership Models*

2.4.1 Maha and Shaikh TCO for open-source software

Shaikh and Conford specifically developed the open-source cost determination model for the United Kingdom government. The government needed to learn the actual cost of the open-source software they used, and thus, they were tasked with developing a model. The model classified the open-source as consisting of factors pushing the software's cost (Maha, 2011). The cost factors are the elements that influence or cause the price of something, in this case software, to increase.

The equation for the model is given as:

$$TCO = \text{direct costs} + \text{indirect costs} \quad (10)$$

The model identified direct cost factors, described as the cost elements that can be accounted for and reflected in the accounting books. The direct costs were the ones that are frequently reflected in accounting details. The other cost factors identified are the indirect costs, which consist of cost elements that may not be inferred directly but influence the cost of open-source software. The expenses were added through the lifecycle of the project.

Owoche (2017) argues that this model is that although it correctly identifies direct and indirect cost factors and is related to open-source software and thus can measure the cost of open-source software, it is specifically not targeted at a software product and, therefore, needs to be more precise. It is general. Another limitation is that the model fails to factor in marketing costs, which is a hidden cost. Thus, the model fails to factor in the hidden costs.

According to Kamau & Sanders (2017), the model cannot be contextualized to the local setting as it was carried out in Europe and Asia. Thus, there is a need to develop an open-source cost determination model that can be contextualized to the local conditions and context.

To solve the identified issues, a refined total cost of ownership model of open-source software must be developed. Böckelman (2017) argues that a better total cost of ownership model should capture hidden costs such as marketing costs and retraining due to users' poor attitudes. The model to be developed should also be specific to a given open-source product so that it achieves its purpose. Moreover, the model has to be contextualized to the local setting for relevancy. Thus, it has to be in the Kenyan setup.

Various researchers argue that research should be based on the local conditions to be relevant. For instance, the factors influencing the cost of open-source software could be different for the African continent than those in Europe (Kibuku et al., 2020). There is also an argument that besides hidden costs, hidden costs such as retraining and marketing of open-source software and employee time should be included in a total cost of ownership model.

There is a limited adoption level of open-source software. This is even higher, as seen by studies in Africa. Despite having no fee, several individuals do not use open-source software but prefer proprietary software. The governments in Africa have the highest adoption levels of open-source software. The reason for not adopting open-source software, among others, is the need for more knowledge of the cost of this software.

*2.5 Theoretical Framework*

Gartner's TCO model was the theoretical framework and basis for developing the new model. Gartner group proposed a model conceptualizing that the total cost of ownership of open-source software is direct cost-plus indirect costs, which gives the total cost of ownership of software. The limitations and weaknesses identified in the models enabled the researcher to develop a conceptual framework factoring in the ideas fronted by the critics of the previous cost estimation methods and models.

*2.6 Conceptual Framework*

The conceptual framework is often a pictorial representation of the variables involved in research and the relationship between them. The representation shown below has the independent and dependent variables, demonstrating their relationship.

The conceptual framework is illustrated below:

## Conceptual framework

### Independent Variables

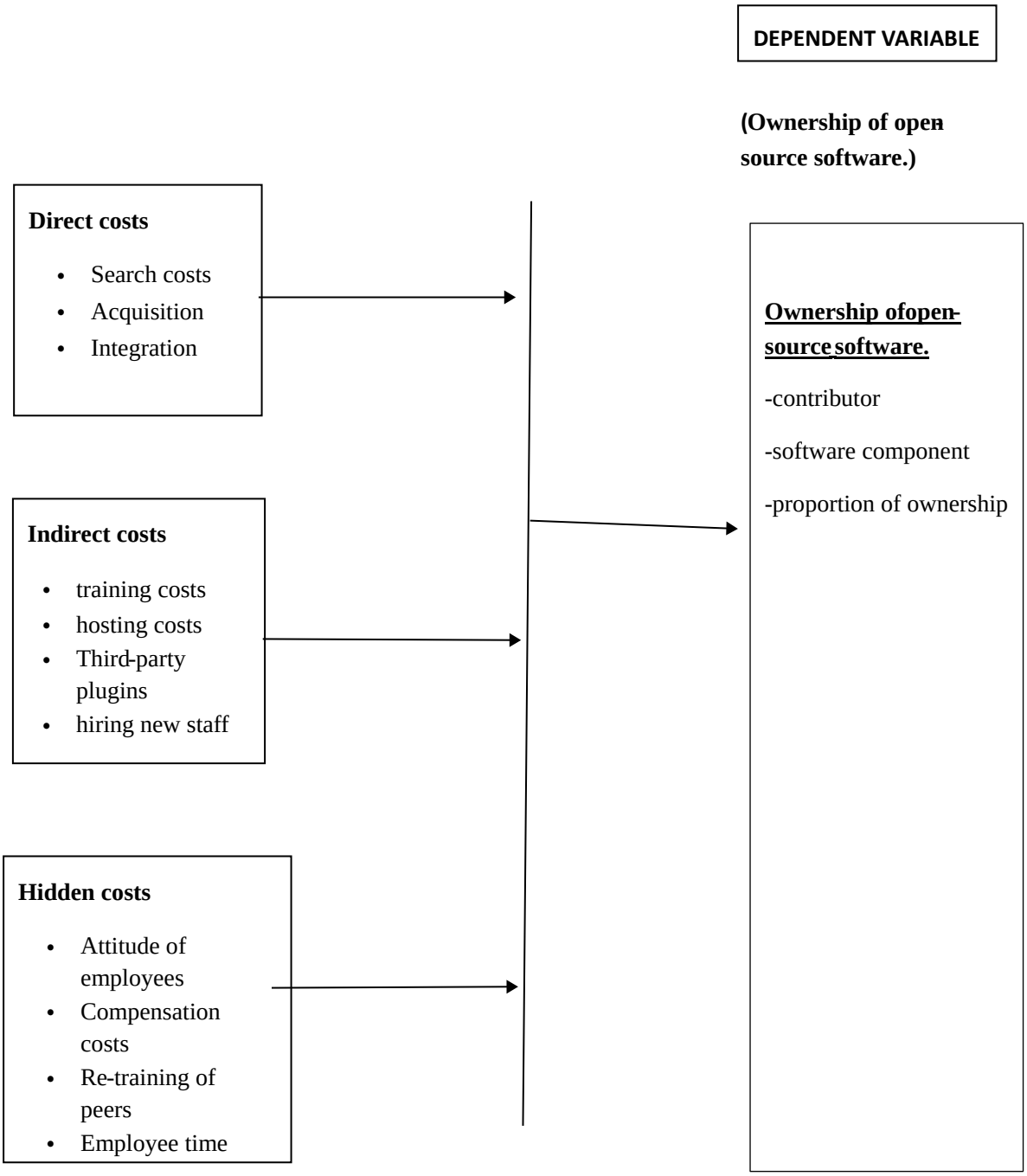


Figure 1. The conceptual framework of the proposed model

*Independent variables* are the variables that are not affected by others, while a dependent variable is one affected by the independent variable. The independent variables in the above representation are direct, indirect, and hidden costs.

The direct costs are the ones that are easily seen and readily accounted for in the lifetime of software (Owoche et al., 2015). The direct costs include search costs, which are the price incurred in the selection of a suitable software; acquisition costs, which entail the cost to put the software in place for it to function correctly; and integration costs,

which are the cost incurred in joining up the software so as it can work properly with the existing systems.

Indirect costs are incurred by software; they are not directly computed but then reflected in the books of account (Owoche et al., 2015). The indirect costs, as depicted in the conceptual framework, are training costs, hosting costs, and third-party plugins needing monthly subscriptions.

Hidden costs, on the other hand, are not easily identified and do not end up in the books of account. The costs entail the attitude of employees, compensation costs, retraining of peers, and employee time.

The dependent variable is the variable whose change is affected by the independent variables. The dependent variable in this is the total cost of ownership of open-source software.

## 3. Research Methodology

The research adopted a multi method methodology to address the study's objectives. The methodology uses quantitative and qualitative data to come up with a solution to a research question. The methodology ensured that the varied objectives of the study were eventually accomplished.

### *3.1 Research Design*

The research design that was adopted by the study was correlational. A correlational design aims to check whether a relationship exists between the variables presented for the study. It helped check whether the independent variables were related to the dependent variable.

### *3.2 Population*

The population for the study was 316 participants in the Open distance and learning departments (ODEL) in public universities in Embu and Kiambu counties.

Table 1. Population for study

| **University** | **county** | **Status** | **Number of participants** |
|---|---|---|---|
| Kenyatta university (Main campus) | Kiambu | Public | 80 |
| Jomo Kenyatta University of science and Technology (**Main campus)** | Kiambu | Public | 74 |
| Mount Kenya University (**Main campus)** | Kiambu | Private | 50 |
| Gretsa University | Kiambu | Private | 45 |
| Embu University (**main campus)** | Embu | Public | 67 |
| | | | **316** |

(Ntarangwi, 2022).

Kiambu County was chosen for study as it possesses the most universities that pioneered adopting an open-source Learning management system (LMS) (Ntarangwi, 2022). 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.

### *3.3 Sample Size Determination*

Mugenda and Mugenda's 30% formula was used to select some participants in the study. The researchers state that for a population of less than 10,000, a 30% representative sample is sufficient (Omona, 2013).

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.

Table 2. 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** |

The sample size chosen was 62 respondents and 15 more to account for any error, making it 77 respondents. Of the 77 respondents, 62 were issued questionnaires, while 15 were interviewed.

### *3.4 Data Collection Techniques*

Questionnaires and interviews collected data for the research. The questionnaire had questions that were on a Likert scale with 5-point values. On the other hand, interviews were structured and answered by experts in the

ODEL departments, such as system administrators of the open-source LMS. The data was collected and analyzed using SPSS version 25.

*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.

## 4. Analysis and Findings

This section provides the results that were found and the analysis of them. The results are presented in the order so as to answer the various objectives.

*4.1 The Direct Cost Factors of Open-Source Software*

The researcher set out the direct cost factors to find out their existence by posing the questions to respondents to ascertain that search, acquisition, and integration costs are direct cost factors. The responses are as shown below.

4.1.1 Search Costs

The search costs are described as the costs incurred when sampling out and deciding on the type of software the institution needs. The responses shown by the histogram below are responses given out by respondents concerning whether they thought search costs existed.

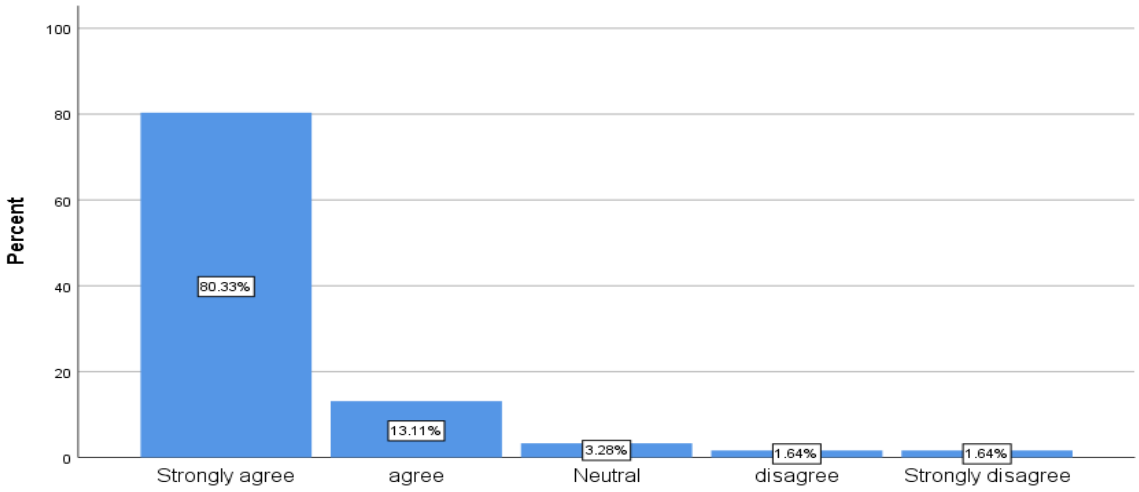


Figure 2. Responses on Search cost

80.33% and 13.11% of respondents strongly agreed and agreed that search costs were incurred when selecting the type of open-source LMS to acquire and use as an institution. Interviews, on the other hand, helped expand on the search costs, whereby it was pointed out that meetings were held to agree on the type of software, expert opinion fee, and fee incurred in demonstrations and presentations.

4.1.2 Acquisition Cost

The acquisition costs are incurred to bring the open-source software searched for and identified by the team to the institution. To find out the certainty of the acquisition being part of the costs incurred, a question was posed to respondents, asking them to give their opinion on whether, indeed, there are acquisition costs. The table below shows the responses given.

Table 3. Responses on acquisition costs

| **Acquisition cost** | | Frequency | Percent |
|---|---|---|---|
| | Strongly agree | 41 | 67.2% |
| | Agree | 16 | 26.2% |
| | Neutral | 3 | 4.9% |
| | Disagree | 1 | 1.6% |
| | Total | 61 | 100.0% |

Table 3 shows that 67.2% and 26.2% agreed and agreed that acquisition cost is incurred in procuring open-source software. This confirms that open-source software accrues acquisition costs.

Upon further analysis of the responses given through interviews, it was revealed that acquisition costs included installation fees and pilot test costs.

4.1.3 Integration Costs

Integration costs are incurred when the system is being installed in the institution. The costs involve ensuring the open-source software's compatibility with the organization's existing systems. The questionnaire was given to respondents, asking them to gauge whether they thought the costs were incurred during the installation of the open-source software. The results obtained are displayed below.

Table 4. Responses on Integration costs

| **Integration costs were incurred** | | Frequency | Percent |
|---|---|---|---|
| | Strongly agree | 42 | 68.9% |
| | Agree | 17 | 27.9% |
| | Neutral | 2 | 3.3% |
| | Total | 61 | 100.0% |

Table 4 shows the responses given by the respondents; it is evident that most of the respondents strongly agreed and agreed at 68.9% and 27.9%, respectively. These confirm that integration costs are incurred when using a new open-source software. The interviews revealed that the integration costs in open-source software are plugins to support the software, APIs acquired to allow communication between the new system and other systems, and the costs incurred in the migration of users.

*4.2 The Indirect Cost Factors of Open-Source Software*

These 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.

4.2.1 Hiring Costs

These are costs that are accrued by an open-source software as a result of it requiring the organization to hire more staff to manage the newly acquired software.

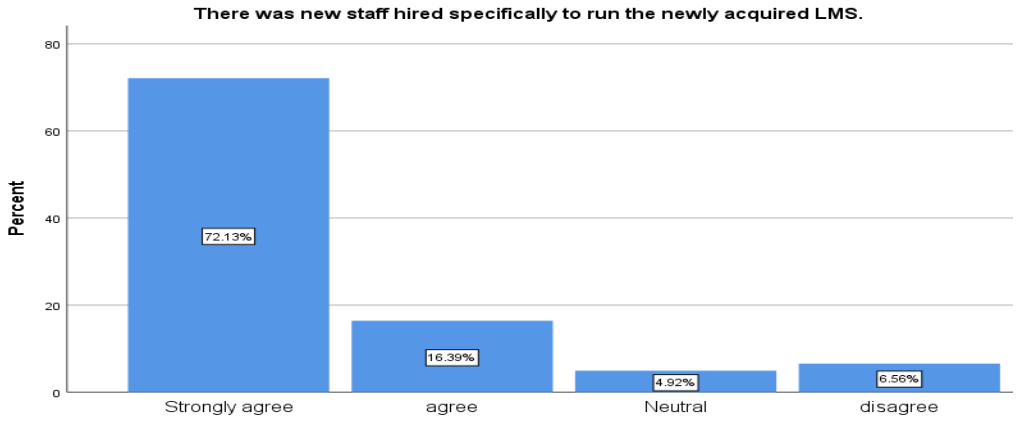


Figure 3. Responses on hiring costs

As shown by the histogram in figure 3 above, most respondents, 72.13% and 16.39%, strongly agreed and agreed. Therefore, hiring costs are indirect to an open-source software product.4.2.2 hosting costs

These are costs that revolve around hosting the open-source software. The results below are from respondents:

Table 5. Responses on hosting costs

| **The open source LMS is hosted and a fee is remitted to pay for it** | | | |
|---|---|---|---|
| | | Frequency | Percent |
| Valid | Strongly agree | 44 | 72.1% |
| | Agree | 10 | 16.4% |
| | Neutral | 5 | 8.2% |
| | Disagree | 2 | 3.3% |
| | Total | 61 | 100.0% |

Table 5 above shows that 72.1% and 16.4% of respondents strongly agreed and agreed, respectively, that open-source software accrues hosting costs. Therefore, hosting costs are indirect costs accrued by open-source software.

4.2.3 Third Party Costs

These are costs incurred in payment to software or services provided by third parties. The question was posed to find out if third party costs are incurred and below are the responses

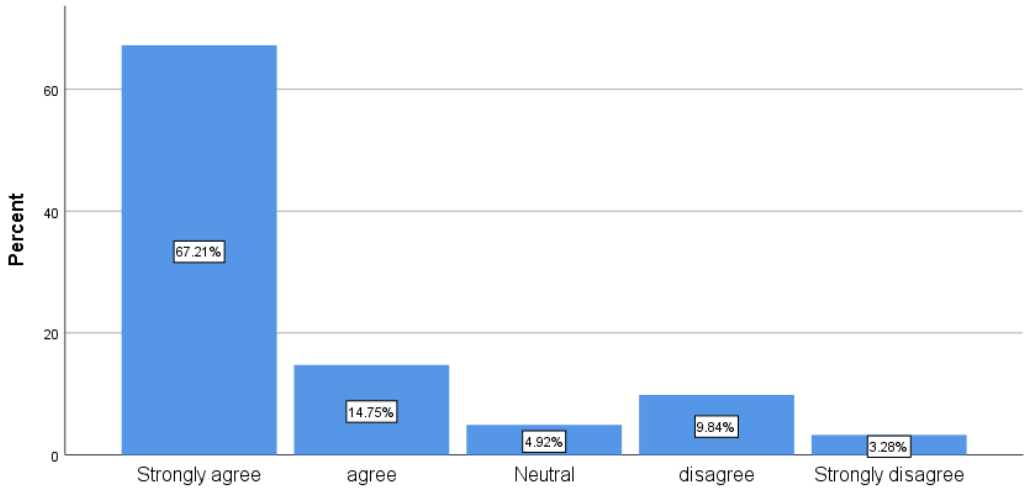


Figure 4. Responses on Third party costs

The histogram majority of respondents, 67.21% and 14.75%, strongly agreed that third-party costs are incurred in open-source software. From the responses, third-party costs are indirect costs incurred.

4.2.4 Compensation Costs

This is money paid out to cover services provided by the staff outside the regular hours or a fee given as a penalty. To ascertain that open-source software incurs compensation costs, a question was posed to respondents, and the answers are given below.

Table 6. Responses on compensation costs

| **Compensation costs are incurred by an open-source software** | | | |
|---|---|---|---|
| | | Frequency | Percent |
| Valid | Strongly agree | 43 | 70.5% |
| | Agree | 9 | 14.8% |
| | Neutral | 3 | 4.9% |
| | Disagree | 5 | 8.2% |
| | Strongly disagree | 1 | 1.6% |
| | Total | 61 | 100.0% |

Most respondents, 70.5% and 14.8%, strongly agreed that open-source software incurs compensation costs. Upon further inquiry, the interviews revealed that the compensation costs included paid time off and overtime pay.

*4.3 The Hidden Cost Factors of Open-Source Software*

The hidden cost factors are costs which are not accounted for and rarely seen as cost factors in an organization.

4.3.1 Marketing Costs

For open-source software to be acceptable in the organization, marketing the product to employees must be done to achieve acceptance and public participation. The respondents were asked whether they thought open-source software incurred marketing costs, and the responses are as presented below:

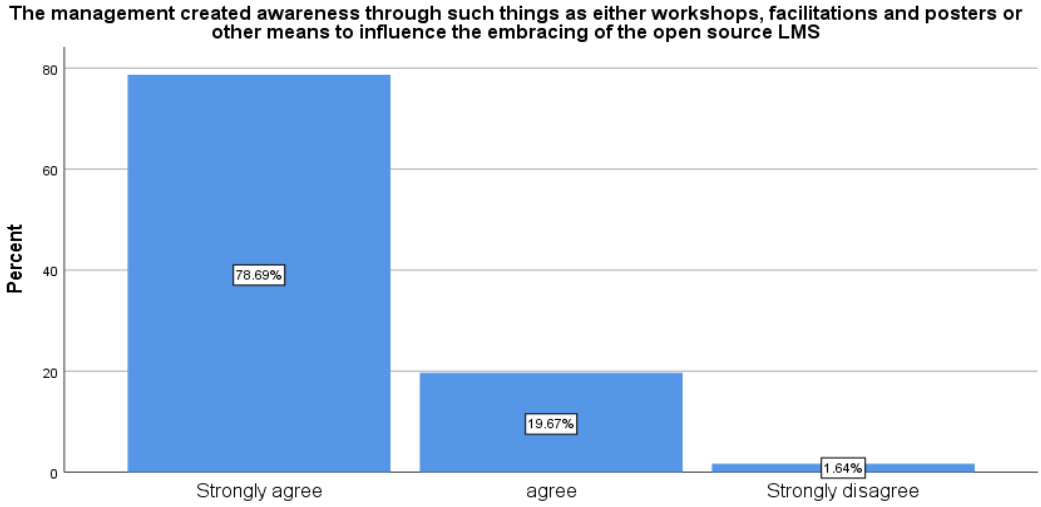


Figure 5. Responses on marketing costs

78.69% and 19.67% of respondents strongly agreed and agreed that marketing costs were incurred in setting up open-source software. The marketing costs are used in workshops, facilitations, posters, and expert-facilitated meetings.

4.3.2 Employee Time

This refers to the extra number of hours that an employee has to work as a result of the introduction of open-source software. This translates to money payable for overtime due to extra hours worked.

To affirm that employee time is a cost to open-source software, a question was posed to respondents, and the result is as shown below:

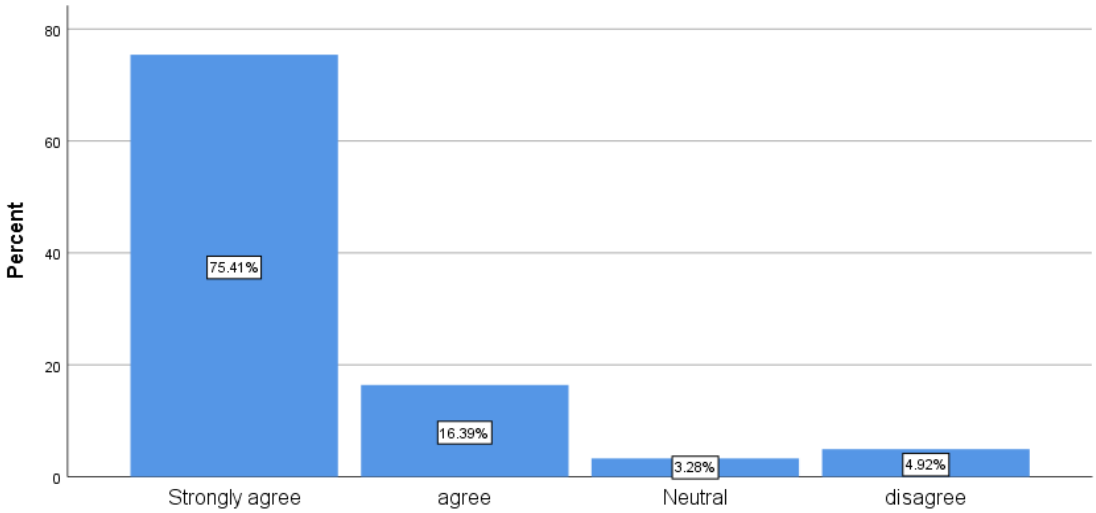


Figure 6. Responses on employee time

75.41% and 16.39% strongly agreed that employee time is a cost in open-source software. The responses thus confirm that employee time is a cost attached to open-source software. Interview responses clarified that employee time includes workload addition costs, system downtime costs, and overtime costs.

4.3.3 Retraining Costs

Retraining is training done again after the official paid-for training has concluded. The retraining takes the form of employing a one-on-one coach for better training or going the extra painful mile of making tutorials. The question was posed to respondents to confirm that retraining is a cost incurred in open-source software. The answers are present in the pie chart below.

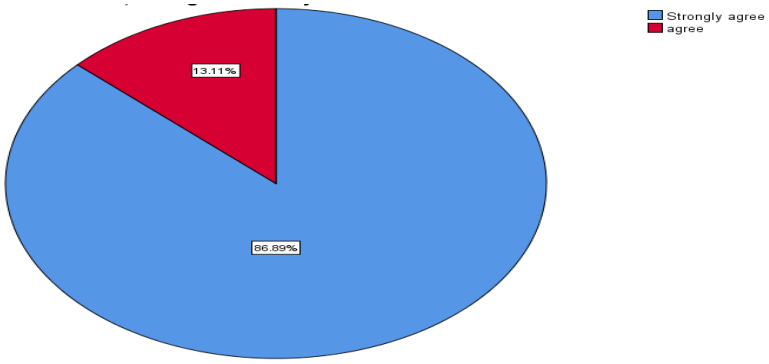


Figure 7. Responses on compensation costs

The pie chart shows that 88.89% and 13.11% strongly agreed and agreed that retraining costs are incurred in the case of open-source software. On further analysis, interviews revealed that the retraining involved the tutorial fee and one-on-one coaching fee.

4.3.4 Attitudinal Costs

These costs result from the attitude of the employees and users of the open-source software. For instance, the resistance to the software may lengthen the learning curve, meaning more time and money.

To find out if attitude is a cost in open-source software, a question was posed to respondents, and the results are as shown below:

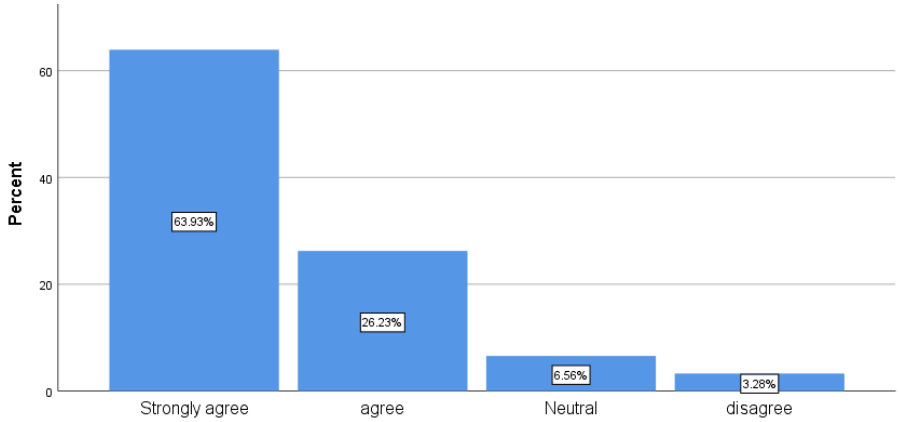


Figure 8. Responses on attitudinal costs

The histogram in figure 8 above shows that 63.93% and 26.23% strongly agreed and agreed that attitude is a cost to open-source software. This confirms that attitude is a cost attached to open-source software. On further inquiry, the interviews shed light on attitude costs: longer-learnability costs and resistance to change costs.

## 5. Relationship between the Independent and Dependent Variables

*5.1 The Relationship between Direct Cost Factors and Total Cost of Ownership of Open-Source Software*

The table 7 presented below was an analysis that majorly examined the correlation between direct cost factors and the total cost of ownership of open-source software. The Spearman's coefficient was used to develop the relationship, as demonstrated in Table 3 below.

Table 7. Relationship between direct factors and total cost of ownership

| Correlations | | | | |
|---|---|---|---|---|
| | | | Total cost_ ownership | Direct_cost_factors |
| Spearman's rho | Total_cost_ownership hip | Correlation Coefficient | 1.000 | .393** |
| | | Sig. (2-tailed) | . | .002 |
| | | N | 61 | 61 |
| | Direct_cost_factors | Correlation Coefficient | .393** | 1.000 |
| | s | Sig. (2-tailed) | .002 | . |
| | | N | 61 | 61 |
| **. Correlation is significant at the 0.01 level (2-tailed). | | | | |

Table 7 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.

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

To establish a relationship between the indirect cost factors and total cost of ownership spearman's correlation was done and shown below are the tables with the results.

Table 8. The relationship between indirect cost factors and total cost of ownership of open-source software

| Correlations | | | | |
|---|---|---|---|---|
| | | | Total_cost_ ownership | indirect_cost_facto rs |
| Spearman's rho | Total_cost_owners hip | Correlation Coefficient | 1.000 | .487** |
| | | Sig. (2-tailed) | . | .000 |
| | | N | 61 | 61 |
| | indirect_cost_fact ors | Correlation Coefficient | .487** | 1.000 |
| | | Sig. (2-tailed) | .000 | . |
| | | N | 61 | 61 |
| **. Correlation is significant at the 0.01 level (2-tailed). | | | | |

The table 8 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 and indirect variables.

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

To establish a relationship between the hidden cost factors and total cost of ownership spearman's correlation was done and shown below are the tables with the results.

Table 9. 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_facto rs | Correlation Coefficient | .594** | 1.000 |
| | | Sig. (2-tailed) | .000 | . |
| | | N | 61 | 61 |
| **. Correlation is significant at the 0.01 level (2-tailed). | | | | |

Table 9 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.

**6. Model Summary**

The model summary presents a summary of the squares and the predictor values and how they relate to the dependent variable. The summary is shown in the table below.

Table 10. The Model Summary

| Model Summary | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | R | R Square | Adjusted R Square | Std. Error of the Estimate | Change Statistics | | | | |
| | | | | | R Square Change | F Change | df1 | df2 | Sig. F Change |
| 1 | .825[a] | .681 | .664 | 1.90099 | .681 | 40.593 | 3 | 57 | .000 |
| a. Predictors: (Constant), Hidden_cost_factors, Direct_cost_factors, indirect_cost_factors | | | | | | | | | |
| b. Dependent Variable: Total_cost_ownership | | | | | | | | | |

Table 10 above, shows a model summary whereby it shows a strong relationship between the predictors which are direct, indirect and hidden cost factors, and the dependent variable total cost of ownership. Total cost of ownership changes are correlated with changes in the predictors, according to the strong positive correlation coefficient (R) of 0.825. The model's predictors account for about 68.1% of the overall cost of ownership variability, according to the coefficient of determination (R Square) of 0.681. This overally shows that there is a strong relationship between the predictors and the dependent variable.

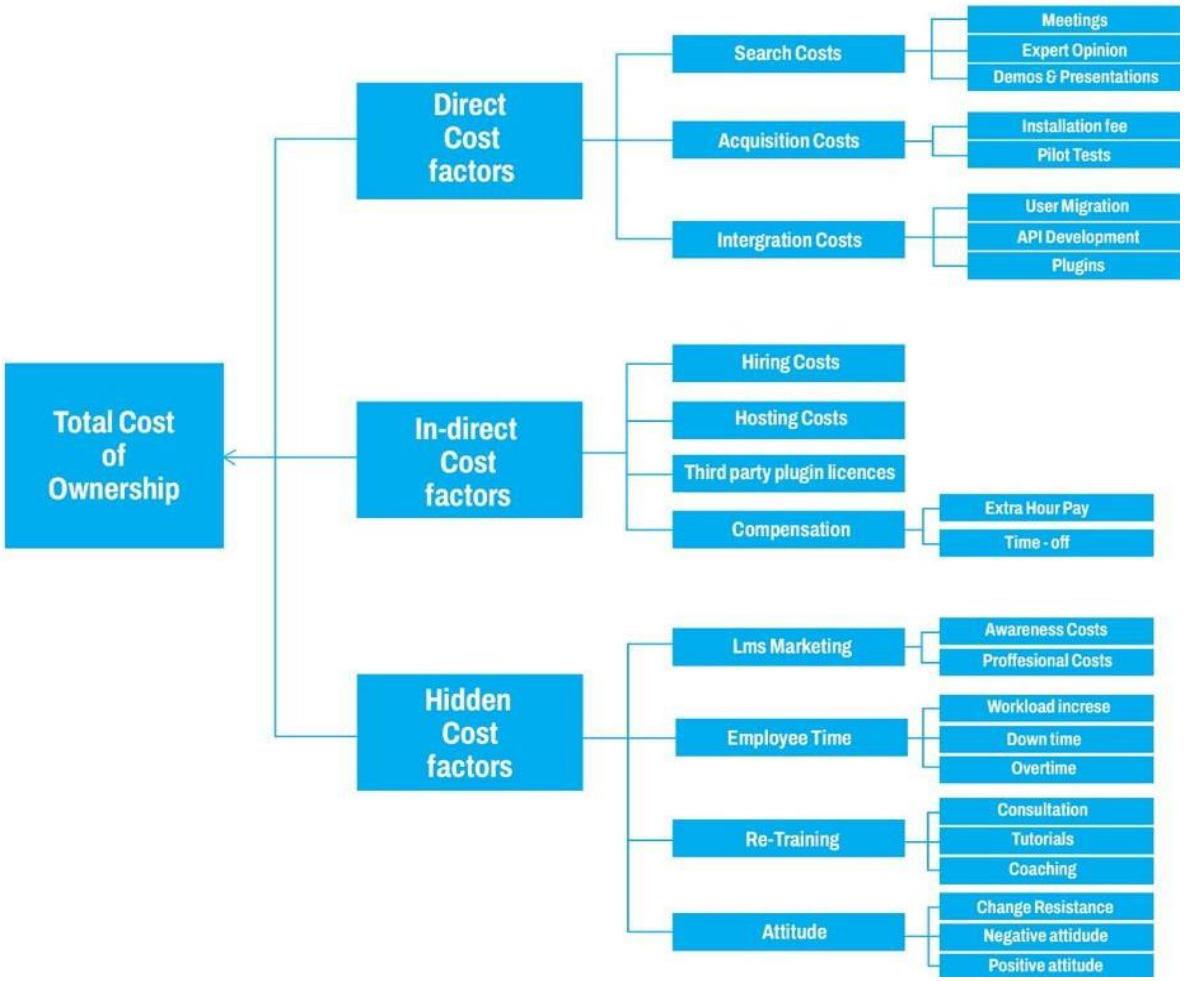


Figure 9. The proposed total cost of open-source software ownership model

The model depicted above in figure shows that the direct, indirect, and hidden cost factors directly cause the cost of ownership of open-source software. The direct cost factors include search costs, which are the initial costs incurred when looking for a product or its vendor; the search costs are further broken down into meetings held, expert opinion fees, and demonstrations and presentations.

The indirect cost factors, on the other hand, include hiring staff, hosting costs, third-party plugin licenses, and compensation, which consist of time-off and extra-hour rate pay.

The hidden costs, on the other hand, 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. 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.

The discussions above show that hidden costs contribute the most to the total cost of ownership of open-source software. The aspect is illustrated perfectly through the mathematical equation derived from the coefficients of the model, as shown below:

**Total cost of open-source software**= 2.000+ 0.134*direct costs + 0.101* indirect costs + 0.430*hidden costs (11)

## 7. Conclusions

In conclusion, the study above explored the problem of software cost determination. A gap was subsequently identified after a review of the available methods used for determining the cost of open-source software. Most of the models pointed out had issues of computing effort alone, capturing only the acquisition cost and ignoring the lifecycle costs of a product. On the other hand, the available total cost of ownership models needed to be more specific to open-source software, thus being insufficient for computing the cost of open-source software.

This study used Gartner's theory of total cost of ownership as the basis for the study. The study used a multi method methodology incorporating qualitative and quantitative methods to collect data. The research design that was applied was a correlational research design collecting data from 77 users of open-source LMS in public universities in Kiambu and Embu counties.

The results showed that the hidden, direct, and indirect cost factors influence the total cost of ownership of open-source software. The hidden cost factors were found to be the most significant contributor to the cost of open-source software. Thereafter a total cost model for open-source software was developed. The model will go a long way in solving cost approximation of open-source software in institutions and various companies.

**Acknowledgments**

I wish to acknowledge Dr. John Wachira Kamau and Dr. Faith Mueni Musyoka for their invaluable contributions to this article. Their expertise, guidance, and support have been indispensable in shaping the content and ensuring its completion. I am deeply grateful for their commitment to this work.

**Authors contributions**

Duncan Kereu Kodhek was responsible for preparing the article, writing the literature review, developing the study design, and collecting and analyzing data. Dr. John Wachira Kamau played a pivotal role in formulating the study topic, revising the article, and served as the primary supervisor for this work. Dr. Faith Mueni Musyoka contributed significantly to data collection, proofreading of the article, and also acted as a supervisor for this project. It is important to note that both Dr. John Wachira Kamau and Dr. Faith Mueni Musyoka made substantial contributions that greatly contributed to the success of this work. All authors have thoroughly reviewed and approved the final manuscript.

**Funding**

There was no funding for this project.

**Competing interests**

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Informed consent**

Obtained.

**Ethics approval**

The Publication Ethics Committee of the Canadian Center of Science and Education.

The journal's policies adhere to the Core Practices established by the Committee on Publication Ethics (COPE).

**Provenance and peer review**

Not commissioned; externally double-blind peer reviewed.

**Data availability statement**

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

**Data sharing statement**

No additional data are available.

**References**

Abdulmajeed, A. A., Al-Jawaherry, M. A., & Tawfeeq, T. M. (2021). Predict the required cost to develop Software Engineering projects by Using Machine Learning. *Journal of Physics: Conference Series*, *1897*(1), 012029. https://doi.org/10.1088/1742-6596/1897/1/012029

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

Bista, G. (2023, January 24). *Total cost modeling of software ownership in Virtual Network Functions*. Theses.hal.science. Retrieved from https://theses.hal.science/tel-04074989/

Böckelman, C. (2017). Cost analysis of cloud based converged infrastructure for a small sized enterprise. *Aaltodoc.aalto.fi*, *28*(12). https://aaltodoc.aalto.fi/handle/123456789/27921

Chi, Y., Dai, W., Fan, Y., Ruan, J., Hwang, K., & Cai, W. (2021). Total cost ownership optimization of private clouds: a rack minimization perspective. *Wireless Networks*, *15*(7). https://doi.org/10.1007/s11276-021-02757-1

Corrado, E. M. (2023). Proprietary and Open Source Software Systems in Libraries: A Few Considerations. *Technical Services Quarterly*, *40*(3), 202-209. https://doi.org/10.1080/07317131.2023.2226434

Gadatsch, A. (2023). IT Investment Calculation and Total Cost of Ownership Analysis: IT Standards as a Tool for IT Controlling. *Springer EBooks*, *16*(6), 75-93. https://doi.org/10.1007/978-3-658-39270-3_6

Gandomani, T. J., Dashti, M., & Nafchi, M. Z. (2022). Hybrid Genetic-Environmental Adaptation Algorithm to Improve Parameters of COCOMO for Software Cost Estimation. *2022 Second International Conference on Distributed Computing and High Performance Computing (DCHPC)*, *9*(4), 82-85. https://doi.org/10.1109/DCHPC55044.2022.9732107

Gollapudi, S., Rath, K., Vrind, T., TY, D., & Rao, P. (2018). A Novel and Optimal approach for Multimedia Cloud Storage and Delivery to reduce Total Cost of Ownership. *EAI Endorsed Transactions on Cloud Systems*, *0*(0), 162596. https://doi.org/10.4108/eai.5-11-2019.162596

Jørgensen, M. (2004). Top-down and bottom-up expert estimation of software development effort. *Information and Software Technology*, *46*(1), 3-16. https://doi.org/10.1016/s0950-5849(03)00093-4

Kahuti, J. (2020, December 21). *Software Cost Estimation – A Comparative Study of COCOMO-II and Bailey-Basili Models*. Ieeexplore.ieee.org. Retrieved from https://ieeexplore.ieee.org/abstract/document/9194166/

Kalantar, K. L., Carvalho, T., de Bourcy, C. F. A., Dimitrov, B., Dingle, G., Egger, R., … Zhang, M. A. (2020). IDseq—An open source cloud-based pipeline and analysis service for metagenomic pathogen detection and monitoring. *GigaScience*, *9*(10). https://doi.org/10.1093/gigascience/giaa111

Kamau, J., & Sanders, I. (2017). *Adoption of Free Desktop Open Source Software in Developing Countries in Africa: A Case of Kenyan University Students*. Retrieved from https://core.ac.uk/download/pdf/154915166.pdf

Kibuku, R., Ochieng, D., & Wausi, A. (2020). 2020. e-Learning Challenges Faced by Universities in Kenya: A Literature Review. *The Electronic Journal of E-Learning*, *18*(2). https://doi.org/10.34190/EJEL.20.18.2.004

Kumar, K., & Behera, H. S. (2020). Software Effort Estimation Using Particle Swarm Optimization: Advances and Challenges. *Advances in Intelligent Systems and Computing*, *14*(8), 243-258. https://doi.org/10.1007/978-981-15-2449-3_20

Link, G. J. P., & Qureshi, S. (2018). The Role of Open Source Communities in Development: Policy Implications for Governments. *Hawaii International Conference on System Sciences 2018 (HICSS-51)*, *21*(12). https://doi.org/10.24251/HICSS.2018.302

Maha, S. (2011). *Maha Shaikh and Tony Cornford*. Retrieved from https://eprints.lse.ac.uk/39826/1/Total_cost_of_ownership_of_open_source_software_(LSERO).pdf

Mthethwa-Kunene, K. E., & Maphosa, C. (2020). An Analysis of Factors Affecting Utilisation of Moodle Learning Management System by Open and Distance Learning Students at the University of Eswatini. *American Journal of Social Sciences and Humanities*, *5*(1), 17-32. https://doi.org/10.20448/801.51.17.32

Mutai, G. K. (2019). *A Framework of Open Source Solution as Tool for Cost Reduction Among Smes in Nairobi: Case of Ainushamsi Energy*. Erepository.uonbi.ac.ke. http://erepository.uonbi.ac.ke/handle/11295/107163

Owoche, P., Gregory, W., & Juma, K. (2015). *Evaluating Total Cost of Ownership for University Enterprise Resource Planning: Case of Maseno University*. Semantic Scholar. Retrieved from https://www.semanticscholar.org/paper/Evaluating-Total-Cost-of-Ownership-for-University-Owoche-Gregory/2df1b4c7549a4405b0dad377ed2c52748ac1a895

Phannachitta, P. (2020). On an optimal analogy-based software effort estimation. *Information and Software Technology*, *125*(8), 106330. https://doi.org/10.1016/j.infsof.2020.106330

Racero, F. J., Bueno, S., & Gallego, M. D. (2020). Predicting Students' Behavioral Intention to Use Open Source Software: A Combined View of the Technology Acceptance Model and Self-Determination Theory. *Applied Sciences*, *10*(8), 2711. https://doi.org/10.3390/app10082711

Rashid, J., Kanwal, S., Wasif Nisar, M., Kim, J., & Hussain, A. (2023). An Artificial Neural Network-Based Model for Effective Software Development Effort Estimation. *Computer Systems Science and Engineering*, *44*(2), 1309-1324. https://doi.org/10.32604/csse.2023.026018

Saleem, M. A., Ahmad, R., Alyas, T., Idrees, M., Farooq, A., Shahid, A., & Ali, K. (2019). Systematic Literature Review of Identifying Issues in Software Cost Estimation Techniques. *International Journal of Advanced Computer Science and Applications*, *10*(8). https://doi.org/10.14569/ijacsa.2019.0100844

Sharma, T., Bhardwaj, A., & Sharma, A. (2011). A Comparative study of COCOMO II and Putnam models of Software Cost Estimation. *International Journal of Scientific & Engineering Research*, *2*(11). Retrieved from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=0fc2259f539200f92e791237a1099fe91887b746

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*, *09*(02), 955. https://doi.org/10.4236/ojbm.2021.92051

Sohaib, O., Naderpour, M., Hussain, W., & Martinez, L. (2019). Cloud computing model selection for e-commerce enterprises using a new 2-tuple fuzzy linguistic decision-making method. *Computers & Industrial Engineering*, *132*(21), 47-58. https://doi.org/10.1016/j.cie.2019.04.020

Srivastava, P., Srivastava, N., Agarwal, R., & Singh, P. (2021). A Systematic Literature Review on Software Development Estimation Techniques. *Second International Conference on Sustainable Technologies for Computational Intelligence*, *9*(12), 119-134. https://doi.org/10.1007/978-981-16-4641-6_11

Tayal, A., Lam, E., Choudhury, D., Dickerson, M., Moovera, G., & Arora, G. (2019). *Determining the Total Cost of Ownership of Serverless Technologies when compared to Traditional Cloud Deloitte Consulting*. Retrieved from http://pages.awscloud.com/rs/112-TZM-766/images/AWS_MAD_Deloitte_TCO_paper.pdf

Wiley, D. (2014). *Bruce Perens, "The Open Source Definition."* Semantic Scholar. Retrieved from https://www.semanticscholar.org/paper/Bruce-Perens%2C-%E2%80%9CThe-Open-Source-Definition%E2%80%9D-Wiley/bd77761fc80f775bc50295251191375db8a27d19