Factors Affecting Software Maintenance Cost of Python Programs

Catherine Wambui Mukunga *cathymukunga@gmail.com*
School of Computing and Information Technology
Murang'a University of Technology
Murang'a, Kenya

John Gichuki Ndia *jndia@mut.ac.ke*
School of Computing and Information Technology
Murang'a University of Technology
Murang'a, Kenya

Geoffrey Mariga Wambugu *gmariga@mut.ac.ke*
School of Computing and Information Technology
Murang'a University of Technology
Murang'a, Kenya

Abstract

One of the primary areas of software project management is cost estimation. The cost estimation problem remains unsolved today because of the ineffective cost estimation techniques which are unsuitable for handling current development methods. Software maintenance costs can be estimated using a variety of models such as the Construction Cost Model (COCOMO), Software Life Cycle Management (SLIM), Software maintenance project effort estimation model and others but more work needs to be done in developing models that can accommodate programs from new programming paradigms. The primary objective of this research was to identify factors affecting the software maintenance cost of python programs and rank them according to their relevance. To achieve the objective, a literature review study was done to identify factors that influence software maintenance costs followed by an expert opinion survey to ascertain which of the factors were relevant for Python programs. Fifty two (52) Python developers and project managers were identified using snowballing technique and asked to rate the cost drivers in order of relevance using a five point scale. Descriptive statistics were used to carry out the analysis of the results. The results indicated that all the eighteen (18) factors affected the maintenance cost of Python programs. The factors were ranked based on the percentage mean of frequencies. Six additional factors were also identified by the experts and ranked. The factors will be considered as input parameters for a cost estimation model to be developed in the near future for estimating the cost of maintaining python programs.

KEYWORDS: Software Maintenance, Cost Drivers, Expert Opinion, Cost Estimation.

1. INTRODUCTION

The importance of software in the 21st century cannot be under estimated, since almost all industries from education, sports, health, and security to governance are enabled by software. Therefore the quality of the software should be of a high level to ensure accurate results (Shrove & Jovanov, 2020). Maintenance is the final step of the Software development process (Aakriti & Shreta, 2015). According to CMRP (2014) corrective maintenance deals with changes to address faults, adaptive maintenance aims to keep the software up-to-date, perfective maintenance ensures the product accommodates proposed requirements and preventive maintenance ensures software is free from failure.

Cost estimation for the maintenance phase is required to determine software reliability, increase productivity, project planning, control, and adaptability. An observation made by Islam and Katiyar (2014) indicates that software maintenance cost is gradually growing with high levels of software life cost relating to the maintenance stage. Controlling the elements that influence software maintenance could aid in cost reduction, predictability, increased productivity, project planning, control, and adaptation of the softwareSingh et al. (2019). According to Pragya and Varun (2012), accurate cost estimation of maintenance can be an estimate of how much resources should be committed to a project and determine the priority of projects. Proper cost estimation also promotes easier management and resource control (Alija, 2017).

Python is described by Chen et al. (2016) as an interpreted and high-level programming language invented by Guido von Rossum with full support for object orientation. Strongly built data structures, binding and linking existing components or services make python suitable for Rapid Application Development and scripting (Saabith et al., 2019). Python is a programming language that enables programmers to effectively connect different sub-systems, promote program modularity and reuse of code (Dev, 2020). Python is a free commercial language that is the 4th most popular out of 100 according to the Importance of Being Earnest index (TIOBE)(index, 2022). Public platforms where programmers learn, collaborate and share code such as GitHub shows how developers largely use Python (github, 2022). Interest in learning python language can also be confirmed through OpenHub which contains volumes of publicly accessible software. One of the most significant bibliographic databases in the academic world Scopus reveals the growing number of applications using the python language.

Though there are several cost estimation models such as the COCOMO, they are unsuitable for the object oriented paradigm (Kukreja & Garg, 2017). It has also been previously argued that COCOMO has some inaccuracies hence the need to revise the model (Obot et al., 2022). Furthermore, python programs are unique from other Object Oriented based programs (Yoo & Lee, 2013) and therefore the existing models cannot precisely predict the maintenance cost of python programs. The work by Alija (2017) has identified software maintenance cost drivers based on a literature review but there is a need to verify and rank the said factors through a primary study.

This work is organized into various sections; section two describes the software cost estimation model COCOMO's versions and the factors influencing software maintenance cost. Section three explains the design and validation of the survey instrument, a procedure for conducting the expert survey and the results. Section four is a discussion of the results obtained from this study. The study is concluded in section five.

2. LITERATURE REVIEW

Factors affecting software maintenance costs were identified by conducting a literature review. A combination of reference list and keywords were adopted as the search strategy. Brereton et al. (2007)Identified seven electronic sources namely; IEEExplore, ACM Digital library, Google scholar, Citeseer library, ScienceDirect, SpringerLink and Scopus as the most relevant sources to Software Engineers. The search covered all studies on software maintenance cost estimation specifically journal papers, conference proceedings and technical reports from the seven sources between the period 2010 and 2022. Factors mentioned in two or more studies were considered for this study.

2.1 Software Cost Estimation Models

Software cost estimation is defined to be a process for predicting development effort (Sangeetha et al.,2012). A study byKeim et al. (2014) defined a software cost estimation model as an indirect measure that predicts the cost of a project with the purpose of budgeting, analyzing risks, project planning and improving the software.

There exist estimation models that implement software maintenance cost influencing factors such as the Constructive Cost Model (COCOMO). The Constructive Cost Model was developed by

Barry Boehm in 1981 and was applied in estimating development cost, man-hours and project schedule as explained in (Boehm, 1983). An earlier Version of COCOMO is the COCOMO 81 which consists of; Basic COCOMO, Intermediate COCOMO and Detailed COCOMO. A later version is COCOMO 2 as mentioned in (Boehm, 1983). Basic COCOMO version can be implemented on organic, semi-detached and embedded project types as explained in (Saljoughinejad & Khatibi, 2018).

Basic COCOMO mathematical statements are presented below as explained in(Boehm, 1983)

Effort Applied (E) = a_b (KLOC)b_b[man-months] (1)

Development Time (D) = c_b (Effort Applied)d_b [months] (2)

People required (P) = Effort Applied / Development Time [count]. (3)

The study by Boehm (1983) explains that the basic COCOMO version was designed to work on estimates from the size of a project which is expressed by kilo lines of code (KLOC). A limitation mentioned by Boehm (1983) is that Basic COCOMO only considers annual change of traffic for maintenance and leaves out factors related to hardware and personnel aspects. A further drawback of the basic COCOMO is mentioned by Boehm et al. (2000) indicating it could not estimate the costs of software from new life cycle procedures and capabilities. Such limitations of the Basic COCOMO led to the intermediate COCOMO model.

The study by Boehm et al. (2000) explains that the intermediate model computes effort from program size in KLOC and four cost drivers, with each driver having some attributes. Attributes affiliated with software products comprise of how operational the software is, the size of database application and ease of product’s understandability and implementation; Hardware elements include; the ability of the analyst, software engineering skills, and a rich experience with programming language while Project attributes include the application of software engineering tools and methods. Boehm (1983) commented that the incorporation of cost drivers in the intermediate model increased its accuracy by 20%. The study by Boehm (1983)has described the Intermediate COCOMO formula to be:

E= a_i (KLoC) (^{b}i) * EAF (4)

Effort (E)is expressed in person-months, size is expressed as Kilo Lines of code (KLOC) and EAF is an effort adjustment factor which is the product of the effort multipliers for each cost driver. Values of aand b depend on the project categories of organic, semi-detached and embedded.

The intermediate COCOMO accommodates sensitivity analysis by altering the ratings of cost drivers.

In an essay reviewed by Bryant & Kirkham (1983) the authors argue that Boehm presented a table of ratings for each cost driver but only alteration of software reliability and some of the personnel attributes are discussed leaving other cost drivers unexamined.

Kitchenham and Taylor(1987) explain that the Detailed COCOMO can counter the drawbacks of the Intermediate model through phase-wise cost driver implementation. The phases of the Detail COCOMO model include planning and requirements, designing of the system, detailed design, module coding and testing and cost constructive model phases as listed in Boehm (2000). The study further explains that effort is computed from program size and cost drivers according to the phases.

Boehm et al. (2009) described the COCOMO II model as a COCOMO 81 update to address software development practices in the 1990s and 2000s. According to Boehm et al. (2009)COCOMO II comprises of the Application Composition Model, which is an early stage model that makes assumptions of systems being developed from components that are reusable, database programming, the early Design Model whose primary purpose is to compute estimations of a project's cost and schedule and the Post Architectural model. COCOMO II consists of cost drivers that are used with the Post Architecture model. The Post Architecture model factors include; required reliability, size of the database, complexity of the product, product reusability, product documentation, time required for execution, storage, hardware and software platforms, personnel analysis and design ability, developer's ability to deal with complex software, personnel turnover, system analyst's capability, level of programming language and software tool experience, software tool application, site collocation and communication support in multisite developments and development Schedule. The rates assigned to the cost drivers are scaled from Very Low to Extra High. Five scale drivers namely; similarity of a product to previously developed products, flexibility of design, design thoroughness and risk elimination, team connectedness and organization's process maturity. The mentioned drivers contribute to a project's duration and determine the exponent used in the Effort Equation. The post architecture COCOMO II model is defined as:

$$PM = A.(Size)1.01 + \sum_{=j=1}^{5} SFj.\prod_{i=1}^{17} EMi \qquad (5)$$

Where B = 1.01+ 0.01×∑SFi and A = 2.45 (Chamkaur et al., 2019). Application of the COCOMO II model in software requirements and maintenance is recommended in the study by(Ismaeel & Jamil, 2007).

COCOMO II model recognizes different approaches such as prototyping, component composition development and database programming. The main focus of COCOMO models was to address the procedural programming paradigm according to (Periyasamy, K & Ghode, 2009) hence an extension of these models would be necessary to accommodate other programming paradigms. In addition according to Obot et al. (2022), COCOMO attributes contain some level of imprecision and therefore there is a need to extend or review the model.

A recent study by Singh (2022) proposed an approach for software maintenance cost using the Putnum model and particle swarm optimization algorithm. Linear Discrement Analysis (LDA) was used for classification. The proposed approach was compared with the COCOMO and Putnum models and use of the Putnum model was reported to contribute to better results.

A study byKyoung-ae-jang and Woo-je Kim(2021)presented a maintenance model for package software. The model is based on maintenance activities identified by reviewing literature and cost structure. Validation of the model was done using real data from maintenance projects and was reported to produce promising results.

A maintenance cost estimation model is developed bySingh et al. (2019) based onthe Tomcat server dataset and particle swarm technique for optimization. The inputs to the model are source lines of code and maintenance effort. The model was said to present more accurate results.

A software maintenance effort prediction model was developed by Maheswaran and Aloysius(2018) based on software cognitive complexity metrics. An empirical study was conducted on the complexity metrics to ascertain whether they were predictors of software maintenance effort.

The work by Islam and Katiyar(2014) proposed a maintenance cost estimation model which is based on technical factors that include; maintenance Staff Ability, internal Complexity, Documentation Quality, testing quality, system life span, code quality, application type, interface complexity, CASE Tools and non-technical factors included understandability, probability, new technology and organization maturity. The maintenance model also operates on fourth generation language environment and applies the annual change of traffic metric.

A study by Kim et al. (2003) proposed a software maintenance project effort estimation model (SMPEEM). Value adjustment factors are introduced and various attributes are discussed such as knowledge of the application area, understanding of programming language and others. Maintenance factors include; quality documentation, Conformance to software engineering standards and testability.

2.2 Factors That Affect Software Maintenance Cost

Research by Chamkaur et al. (2019) identified software cost factors and categorized them under technical and non technical factors. Technical factors listed included; complex Software, Human capability, quality of documentation, Configuration management technology, Modern Programming Specifications, size of the Database, Component Reusability and Component Performance. Non-Technical Factors: Application Experience, Staff Stability, External environment, Support environment and User needs. The researchers recommended the implementation of the factors in reducing maintenance costs by the use of a cost estimation model.

A study by Alija (2017) has highlighted team cohesiveness, contractual responsibility, staff capability, program duration and composition, costs of understanding or program comprehension, lacking or incomplete documentation and impact analysis as justification for increasing software maintenance costs.

A study by Balra (2017) presented a list of factors affecting software maintainability such as understandability, standards, modularization, and the language of a program, testing, complexity and traceability.

Research by Benaroch (2013)sought to study the work and contribution by considering application characteristics and personnel attributes as software maintenance cost drivers. The factors include System characteristics namely; system age, size of the system and software complexity. Personnel factors (independent) include; diversities of location and skills and the maintenance personnel. System Dependent factors include maintenance effort and maintenance cost. The research concluded that personnel attributes have a larger influence than system factors.

Dehaghani and Hajrahimi (2013) carried out a study to identify factors that had a greater effect on software maintenance costs. Interviews were conducted and factors were identified using Analytic Hierarchy Process (AHP) and prioritized using expert choice (EC) software. The study mentioned factors such as the reliability of software, the size of a system, the complexity of the system, required time for execution, storage requirements, the experience of the programmer with a programming language, quality of documentation and others.

In the work by Pragya and Varun (2012) cost estimation factors of component based software are discussed. The factors are categorized as technical and non-technical. Factors listed under technical factors include; component efficiency, application's ease of use, the interaction of system sub-elements, reusability of system's units of composition, application's connection, ease of product use, ease of product maintenance, user training, computer-aided tools and interface complexity. Non-technical factors include; expert experience, requirements stability, technology advancements, system components and organization progressive improvement. The study by Pragya and Varun(2012) presented a COCOMO based cost estimation model for maintaining component based software. The model is based on software development cost, the amount of source code changing within a year and high-tech and low-tech factors that have an effect on component based software.

A study was conducted by Lee (2011) involving the identification of factors for estimating effort of conducting corrective maintenance of object oriented programs. The factors were grouped into; developer-related factors, environmental factors, defect factors and code factors. Developer factors included familiarity with technology and software product and low system development experience. Environmental factors included; tool unavailability, minimal team cohesiveness and others. Defect factors included; unavailability of bugs documentation, minimal defect reproduction, codebase's ineffectiveness at the start of a maintenance project, Code related factors included; high code complexity, low maintainability of code structure, high level of code/ system dependencies, high version/deployment complexity, high level of code volatility and low availability of formal design documentation and code comments.

Software maintenance cost factors are discussed in Yong chang et al. (2011) and categorized as technical and un technical attributes. Technical factors include; the complexity of software, human capacity, quality of program documentation, configuration management technology, modern programming specifications and database size. Untechnical factors included; experience with the application, staff low turnover, time to develop the application and others. The factors were assigned weight values and implemented in the calculation of maintenance cost using an empirical method.

Based on the literature review, eighteen factors were identified as summarized in table one

	Factor	**Description**	**References**	**Number of studies**
1	Software Complexity	Defines the complexity of internal properties of a software	[4][5][6][7] [8] [9] [10] [11]	8
2	Document Quality	Degree of correctness and completeness of system documents.	[4] [6] [7] [8] [11]	5
3	Configuration Management Technology	The technology used in the maintenance of computer systems.	[4] [11]	2
4	Modern Programming Specifications	Involves using modern tools and techniques for program maintenance.	[4][7] [11]	3
5	Program Size	Size is measured by counting code lines in a program, counting the number of classes and functions.	[4] [5] [7] [11]	4
6	Component Reusability	The extent to which various system components can be reused. Reuse will lower maintenance costs.	[4] [9]	2
7	Component Performance	The efficiency of various components incorporated in the system.	[4] [9]	2
8	Maintenance Staff stability	Describes the permanency of the maintenance team.Frequent staff turnovers could result in more time to understand the system.	[4] [6] [11]	3
9	Testing Quality	The quality of tests on the system. Low quality tests will attract higher maintenance costs.	[6] [10] [11]	3
10	System Lifespan	Age of the system under maintenance	[6]	1

11	Application Type	An application that is well understood will result in minimal change requests.	[5] [6]	2
12	CASE tools	The use of various Software Engineering tools imply the cost of maintaining software	[6] [8] [9]	3
13	Dependence on External Environment	An application would need modification if it is dependent on the external environment,	[4] [8] [11]	3
14	Hardware Stability	The stability of a hardware configuration on which software will operate on.	[7]	1
15	Programming Style	Style of programming is defined by guidelines and rules to follow in writing a computer program.	[7] [11]	2
16	Understandability	How easy it is to comprehend applications.	[6] [8] [10]	3
17	Technology Newness	Refers to how new the technology being implemented is and the frequency of technological changes. New technology might require training or hiring skilled personnel.	[6] [9]	2
18	Organization Maturity	A Measure of an organization's readiness and capability to conduct software maintenance.	[6] [9]	2

TABLE 1: Factors affecting software maintenance cost.

A total of eleven studies were reported to contain factors influencing software maintenance cost. Software complexity and document quality were the most mentioned factors. System lifespan and hardware stability were the least mentioned in one study each.

3. EXPERT OPINION SURVEY

Baker et al. (2014) defined expert elicitation as the extraction and quantification of personalized opinions in research.

This section presents subsections explaining the design and validation of the survey instrument and how the expert opinion survey was conducted. Survey results are also presented in three sections namely; demographic survey of respondents', responses on software maintenance cost factors and ranking of the factors.

The expert opinion process followed the steps defined by Mosleh and Apostolakis (1987) namely; the definition of the problem statement, the selection of experts using a set criteria, informing experts of the objectives and expectations of the study and decision making. This research used a survey research design, questionnaires were used for data collection and data analysis was done using SPSS version 20. The research implemented a deductive approach. The experts were selected using snowballing technique where a few experts were identified who later identified fellow experts until there was an acceptable number of respondents. A requirement was that an individual had to have worked for three or more years as a python developer or in python project management to be classified as an expert. An online survey was carried out and responses were analyzed using descriptive statistics.

3.1 Design and Validation of Survey Instrument

An expert opinion questionnaire was designed for this study and contained two sections. The first section labelled A comprised of respondent's personal information and second section labelled B contained software maintenance cost factors. A Likert scale of five was used to rate the factors namely; 1. Very low, 2. Low, 3. Slightly high 4. High and 5. Very high. The factors were divided into two categories of technical and non-technical. Technical factors consisted of; software complexity, document quality, configuration management, modern programming specifications, program size, component reusability, component performance, maintenance staff stability, testing quality, system lifespan, application type, CASE tools, dependence on the external environment, hardware stability and programming style. Non-technical factors were; understandability, technology newness and organization maturity. The responses of the pilot study were excluded from the analysis of the final results. Content validity was performed on the questionnaire to assess the extent to which measurement instrument items are relevant and representative of the target construct. A total of eight experts were randomly selected to test the questionnaire items on their degree of relevance and degree of clarity.

The experts were guided by the following scales:

Degree of item relevance to the measured domain
1- Not relevant
2- Somewhat relevant
3- Quite relevant
4- Highly relevant

Degree of item clarity
1- Unclear
2- Needs revision
3- Clear with minor revision
4- Very clear

Responses from the experts were tabulated and computed in MS Excel and yielded a score content validity index of 0.988889 for the degree of relevance and 0.983333 for the degree of clarity which according to Shi and Sun(2012) is within the acceptable value of 0.78 or higher. Cronbach's alpha was used to assess the internal consistency of the questionnaire. The measure was applied because it can measure a survey consisting of multiple Likert-type scales and items. A Cronbach's Alpha value of 0.814 was recorded. According to Mohsen(2011), satisfactory values of alpha are between 0.70 and 0.95 hence the instrument for data collection was considered reliable. Cronbach's alpha was applied because it is easier to use in comparison to other estimates and only requires one test administration (Mohsen, 2011). Questionnaire reliability statistics are presented in table two.

Reliability Statistics

Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	No. of Items
.814	.810	18

TABLE 2: Questionnaire Reliability Statistics.

Based on the above values, it was concluded that the questionnaire had achieved a satisfactory level of content validity.

3.2 Conduct of Expert Opinion Survey

The goal of the survey was to evaluate the relevance of factors contributing to software maintenance costs. Experts were identified by snowball technique and issued an online questionnaire. Developers and project management staff with over three years in Python project development and maintenance were selected. The sample size comprised of 52 python experts. The Survey Planet platform was used to host the study questionnaire. Response time for the survey was four weeks, the respondents were not contacted on the same day which explains the overall long period of the online survey.

According to Naderifar et al. (2017) sampling is done until data saturation in snowball method. Experts were familiarized with the goal of the elicitation process and the details of the issues involved. The respondents were provided with the list of software maintenance cost factors that were identified in the literature and were required to rank the factors based on the Likert scale. In addition the experts were required to include other factors not in the list and influenced software maintenance cost.

3.3 Survey Results

Feedback from the respondents was received and the data collected were analyzed using SPSS software.

a) Demographic Summary of the Respondents

Characteristics of the respondents such as ooccupation, years of experience in cost modelling and estimation, years of experience with Python software and level of education were considered.

i. Respondents Occupation

Analysis of respondents' occupations was done and the findings indicated 28.6% of the respondents were involved in software project management while 71.4 % were in software development. These findings were acceptable because all the respondents had knowledge and experience in python software development and software project management. The findings are presented in figure 1.

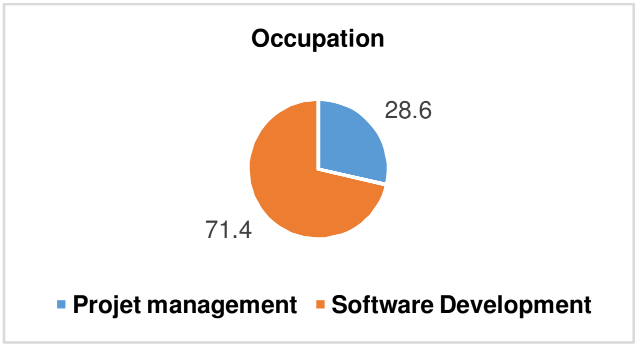

FIGURE 1: Responses on Respondent's occupation.

ii. Software Modeling and estimation experience of the Respondents

An analysis of the respondents' experience in Software modelling and estimation was carried out. Findings indicated that two respondents constituting 4.8% had less than two years experience however they still qualified as experts since they had five and six years of python software development. 73.8% of the respondents had three to five years experience, 16.7% had six to ten years experience and 4.8% were highly experienced with over ten years in software modelling and estimation. Findings are presented in figure 2.

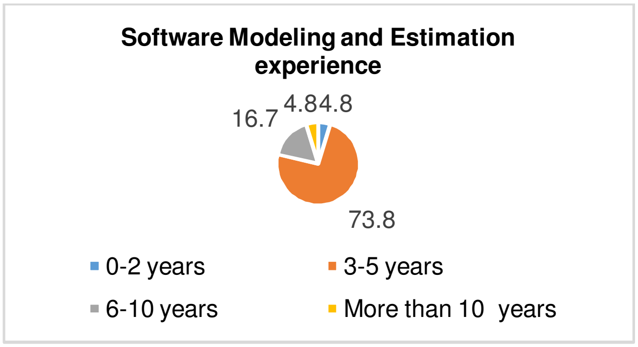

FIGURE 2: Responses on Software Modelling and Estimation experience.

iii. Python Language Experience

Analysis of respondent's responses on python language experience was conducted and the findings indicated that none had less than three years experience. 61.9% had three to five years experience, 31% was in the category of six to ten years experience and 9.5% had over ten years experience with python applications. The findings were a clear indicator that the respondents were qualified to participate in the study and would provide reliable information. Findings are presented in figure 3.

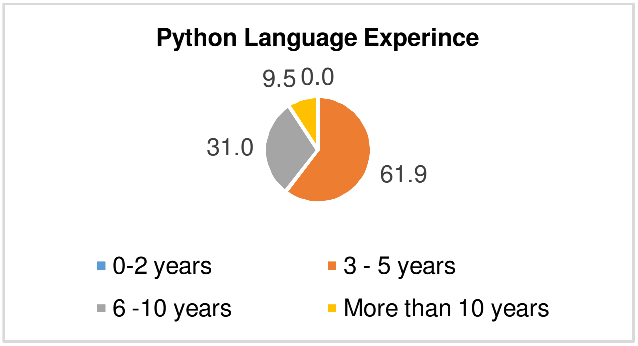

FIGURE 3: Responses on Python Language Experience.

iv. Academic Qualifications

An analysis was conducted of respondents' academic qualifications. The findings indicated that 4.8% of the respondents had a Diploma, 57.1% of the respondents had a Bachelor's degree, 31% had a Master's degree and 7.1% had a Ph.D. degree. Two respondents with Diploma academic qualifications had five years of experience in developing python applications and three years experience in software modelling and estimation hence were considered as experts. Findings on academic qualifications are presented in figure 4.

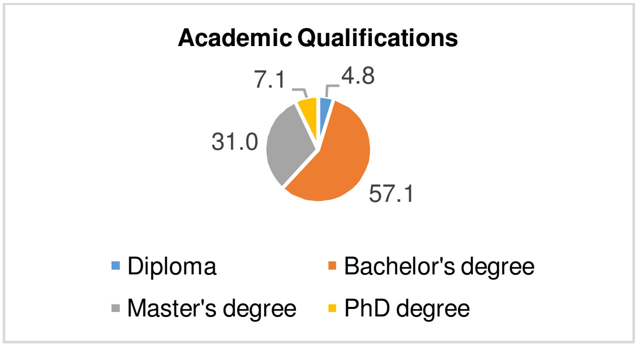

FIGURE 4: Responses on Academic Qualifications.

b) Expert opinion Results on Software Maintenance Cost Factors

The respondents rated the factors based on how much they influence maintenance cost using a scale of very low to very high. A rating of very low means the factor's influence on the cost of maintenance is very low and very high means the factor has a lot of influence on software maintenance cost.

i. Ranked Software Maintenance Cost Factors

Each factor was ranked based on the mean value to identify the most relevant. The Sum of actual responses was divided by the Sum of expected responses to get the mean using the formula shown below;

$$Mean = \frac{Sum\ of\ actual\ responses\ \ per\ attribute}{Sum\ of\ expected\ responses\ \ per\ attribute} \times 100 \qquad (6)$$

The Sum of actual responses was derived by summing the respondents' actual responses on all the items in the questionnaire. The Sum of expected responses per attribute was derived by summing expected responses from all the items in the questionnaire. Each item's expected response was five on the scale provided.

From the analysis, understandability, quality of documents, technology for configuration management, programming standards, program size and complexity of software were highly ranked with over 80% mean values. CASE tools, application type and system life span had the lowest means of less than 60%.The results are presented in table 3.

Ranking	Factors affecting maintenance cost	Sum of Actual Responses	Sum of Expected Responses	Mean in %
1	Understandability	180	210	85.71
2	Document Quality	174	210	82.8
3	Configuration Management Technology	173	210	82.3
4	Modern Programming Specifications	170	210	80.95
5	Program Size	170	210	80.95
6	Software Complexity	168	210	80
7	Testing Quality	167	210	79.52
8	Component Reusability	165	210	78.57
9	Organization Maturity	163	210	77.61
10	Maintenance Staff Ability	162	210	77.14
11	Technology Newness	155	210	73.80
12	Programming Style	146	210	69.52
13	Hardware Stability	142	210	67.61
14	Dependence on External Environment	134	210	63.80
15	Component Performance	133	210	63.33
16	CASE Tools	124	210	59.04
17	Application Type	117	210	55.71
18	System Lifespan	116	210	55.23

TABLE 3: Ranked software maintenance cost factors.

Additional factors that influence the cost of maintaining Python programs mentioned by the experts included; availability of maintainers, Staff ability and skills, code quality, number of maintainers, hiring model of maintainers and location diversity of maintainers. A second questionnaire was issued to the experts to get their feedback on how relevant the factors were in the determination of software maintenance cost.

Each of the additional factors was ranked based on the mean value to identify the most relevant. The Sum of actual responses was divided by the Sum of expected responses to get the mean. The Ranked factors are presented in table 4.

From the analysis code quality was ranked highest with 85.8 % mean. Location diversity of maintainers was the least relevant factor with a 67% mean.

Ranking	**Factors affecting maintenance cost**	**Sum of Actual Responses**	**Sum of Expected Responses**	**Mean in %**
1	Code quality	73	85	85.8
2	Staff ability and skills	68	85	80
3	Availability of maintainers	64	85	75.2
4	Number of maintainers	59	85	69.4
5	Hiring model of maintainers	58	85	68.2
6	Location diversity of maintainers	57	85	67

TABLE 4: Ranked software maintenance cost factors.

4. DISCUSSION

A deductive research methodology was adopted with results obtained from this study comprising a mix of secondary and primary data. Eighteen factors were identified from the literature review namely; Understandability, quality of documents, technology for managing software Configurations, Programming standards, Program size, the complexity of Software, Testing quality, Component reusability, Organization maturity, Maintenance staff ability, Technology newness, Programming style, Hardware stability, Dependence on the external environment, Component performance, CASE tools, Application type and System lifespan.

An expert opinion survey was conducted to enhance and verify the findings from the literature review. All the respondents had over three years experience in python language and hence were considered as experts. Over 95% of the respondents had a minimum academic qualification of a Bachelor's degree. Such findings on the demographic characteristics of the respondents indicated that they were fit for the study. From the expert's opinion survey, all the factors were reported to have some influence on the maintenance cost of Python programs. However, the degree of relevance was differentiated by the ranking done based on the percentage mean per factor. Understandability, document quality, configuration management technology, modern programming specifications, program size and software complexity were top on the list with over 80% mean values in the first survey. CASE Tools, Application Type and System Lifespan were the least significant with less than 60% mean.

In the second survey code quality and staff ability and skills were the most influential factors with over 80% mean while location diversity of maintainers was the least significant.

Previously, the identification of software maintenance cost drivers has relied on literature review as seen in Alija (2017) and frameworks as presented in the work by Benaroch (2013). Very few authors have taken the time to conduct primary studies to verify the factors. This work has made a contribution by verifying the factors through an expert opinion survey and in addition, ranking the factors. Ranking will provide the beneficiaries of this work with prioritized factors during software maintenance cost estimation.

5. CONCLUSION AND FUTURE WORK

This research aimed to identify and present a set of maintenance cost drivers for python programs and show the order of relevance. This was done through an expert opinion survey to determine relevant factors for estimating the maintenance cost of python programs. A total of twenty four factors were reported and ranked in order of relevance. Adoption of the mentioned factors by project managers can greatly impact decision making and success in the estimation of software maintenance costs. This work has also a great benefit to software developers since it presents information for developers to understand which factors affect maintenance cost and their ranking in terms of relevance. A metrics-based neural-fuzzy cost estimation model will be developed in the near future and the updated factors will be considered as input parameters. Incorporating updated cost drivers will potentially result in a more accurate estimation model for python programs.

6. REFERENCES

Aakriti, & Shreta. (2015). Software Maintenance challenges and issues. *International journal of computer science engineering, 4* (1), 23-25.

Alija. (2017). Justification of software maintenance costs. *International Journal of Advanced Research in Computer Science and Software Engineering, 7* (3), 15-23.

Balraj, K. (2017). A Survey of Key Factors Affecting Software Maintainability. *International Journal for Research in Applied Science & Engineering Technology, 5* (6), 1631-1637.

Benaroch. (2013). Understanding Factors Contributing to the Escalation of Software Maintenance Costs . *Thirty Fourth International Conference on Information Systems.* Milan.

Boehm. (1983). Software Engineering Economics. *ACM, 8* (3), 44 - 60.

Boehm, Abts , & Chulani. (2000). Software development cost estimation approaches - a survey. *Annals of Software Engineering, 10*, 177 - 205.

Boehm, Abts, Brown , & Chulani. (2009). *Software cost estimation with COCOMO II.* Prentice Hall Press.

Boehm, Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., & Steece, B. (2009). *SSoftware cost estimation with COCOMO II.* Prentice Hall Press.

Brereton, Kitchenham, Budgen, & Turne. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of systems and software, 80* (4), 571 - 583.

Bryant, & Kirkham. (1983). B .W.BOEHM SOFTWARE ENGINEERING ECONOMIC S A REVIEW ESSA Y. *ACM , 8* (3), 44.

Chamkaur, Neeraj, & Narender. (2019). Analysis of software maintenance cost affecting factors and cost models. *International journal of scientific and technology research, 8* (9), 276-281.

Chen, Chen, Ma, & Xu. (2016). Detecting code smells in python programs. *International conference on software analysis testing and evolution.* Nanjing,China.

CMRP. (2014). *Maintenance engineering handbook.* McGraw-Hill Education.

Dehaghani, S. M., & Hajrahimi, N. (2013). Which factors affect software projects maintenance cost more? *Acta Informatica Medica, 21* (1), 63-66.

Dev. (2020). Design and development with Python programming. *Journal of Engineering & Technology*, 26-30.

github. (2022). Retrieved from Github.com: Github.com

index. (2022, November 12). Retrieved from tiobe.com: https://www.tiobe.com/tiobe-index/

Islam, & Katiyar. (2014). Development of a software maintenance cost estimation model: 4th GL perspective. *International Journal of Technical Research and Applications , 2* (6), 65-68.

Ismaeel, & Jamil. (2007). Software engineering cost estimation using COCOMO II model. *Al-Mansour J, 10*, 86-111.

Keim, Bhardwaj, saroop, & Tandon. (2014). Software Cost Estimation Models and Techniques: A Survey. *International Journal of Engineering Research & Technology (IJERT), 3* (2), 1763 - 1768.

Kim, H., Kim, S., Suh, J., & Ahn, Y. (2003). The software maintenance project effort estimation model based on function points. *JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE, 15* (2), 71-85.

Kitchenham, & Taylor. (1987). Software Cost Models. *ICL Technical Journal, 4* (1), 73-101.

Kukreja, & Garg. (2017). Effort estimation of object orinted system usig Stochastic tree boosting technique. *International journal of advanced research in computer science*, 91-96.

Kyoung-ae-jang, & Woo-je Kim. (2021). A method of activity-based software maintenance cost estimation for package software. *The Journal of Supercomputing* , 8151 - 8171.

Lakens. (2022). Sample Size Justifcation. *Collabra: Psychology, 8* (1), 1-32.

Lakens, D. (2021). Sample size justification. *psyarxiv*.

Lee, M. J. (2011). Identifying effort estimation factors for corrective maintenance in object-oriented systems. Las Vegas: digitalscholarship.unlv.edu.

Leung, & Fan. (2002). *Software cost estimation. In Handbook of Software Engineering and Knowledge Engineering.*

Maheswaran, & Aloysius. (2018). Empirical Validation Of Object Oriented Cognitive Complexity Metrics Using Maintenance Effort Prediction. *International Journal of Scientific Research in Computer Science Applications and Management Studies*.

Mohsen. (2011). Making sense of Cronbach's alpha. *International Journal of Medical Education, 2* (1), 53-55.

Mosleh, & Apostolakis. (1987). The elicitation and use of expert opinion in risk assessment: a critical review. *Probabilistic safety assessment and risk management, 1* (PSA 87).

Naderifar, Goli, & Ghaljaie. (2017). Snowball sampling: A purposeful method of sampling in qualitative research. *Strides in Development of Medical Education, 14* (3), 1-4.

Periyasamy, K, K., & Ghode, A. (2009). Cost estimation using extended use case point (e-UCP) model. *International Conference on Computational Intelligence and Software Engineering.*

Pragya, S., & Varun, K. (2012). A Cost Estimation of Maintenance Phase for Component Based Software. *Journal of Computer Engineering, 1* (3), 1-8.

Saabith, Fareez, & Vinothraj. (2019). Python current trennd applications - an overview. *International Journal of Advance Engineering and Research Development*, 6-12.

Saljoughinejad, & Khatibi. (2018). A New Optimized Hybrid Model Based on COCOMO to Increase the Accuracy of Software Cost Estimation. *Journal of Advances in Computer Engineering and Technology, 4* (1), 27-40.

Sangeetha, Latha, & Prasad. (2012). software Cost Models. *International Journal of Engineering Research & Technology (IJERT)*, 1-10.

Shi, J. M., & Sun, Z. (2012). Content validity index in scale development. *Journal of Central South University. Medical sciences*, 152-155.

Shi, J. M., & Sun, Z. (2012). Content validity index in scale development. *Journal of Central South University. Medical sciences, 37* (2), 152-155.

Singh, Kamini, Juneja, Joshi, & Garg. (2022). Performance comparison of Putnam model using new technology trends for software maintenance cost estimation. *Journal of Discrete Mathematical Sciences and Cryptography* , 691-703.

Singh, Sharma, & Kumar. (2019). An Efficient Approach for Software Maintenance Effort Estimation Using Particle Swarm Optimization Technique. *International Journal of Recent Technology and Engineering (IJRTE)*, 1-6.

Singh, Sharma, & Kumar. (2019). Analysis Of Software Maintenance Cost Affecting factorsand estimation models. *International journal of scientific & technology*, 276-281.

Yongchang , R., Tao , X., Xiaoji , C., & Xuguang , C. (2011). Research on Software Maintenance Cost of Influence Factor Analysis and Estimation Method. *IEEE*.