SOFTWARE COST DRIVERS AND COST ESTIMATION IN NIGERIA

ASIEGBU B, C AND AHAWE, J

Abstract

This research work investigates the effect of cost drivers on software cost estimation. Several models exist that provide one or more mathematical algorithms that compute cost as a function of a number of variables or cost drivers. There are also several other tools and techniques available to assist in creating software project cost estimates, they are still very inaccurate. Several authors in software development have given varied suggestions for these inaccuracies and ways to overcome some of them. All classes of technique are challenged by the rapid pace of change in software technology. The purpose of this research is to determine the extent to which software cost variables affect cost of software development. Primary data that were collected by Likert scale were analyzed using Multiple Regressions tools. The first hypothesis in this work was tested using multiple regression analysis tools. Results from this work led to the conclusion that the cost drivers as a whole were contributors in predicting cost of software development in Nigeria.

Keywords: Software Cost Drivers; Cost Estimation; Nigeria

Research Background

Information technology practitioners are worried over poor records of implementing information technology projects in Nigeria. There is hardly any information technology project that does not end in millions of naira over and above the estimated amount, (Adibe, 2003). In addition to scope and time, cost is an essential component of project management. Cost has a direct relationship with scope and time. Poor project cost estimation sometimes leads to project failure. More worrisome is the subsystem called software. Managing software development project is difficult and different from managing other projects such as research projects that has a high degree of uncertainty. Software development project does
not have such level of risk, yet the history of project management in software arena has a poor success rate. According to Adibe, (2003), “Most software development projects in Nigeria (numbers as high as 85% have been quoted) fail to deliver the system to the original or user’s requirement and within the original time and budget predictions”. Agreed that the problem is not peculiar with Nigeria, but the frequency at which it occurs is alarming and calls for urgent attention. According to Suhanic (2001), the average cost overrun – the additional percentage or dollar amount by which actual costs exceed estimates for information technology projects in America was 189 percent of their original estimates. More than 31 percent of software projects were canceled before completion, costing American companies and government agencies billions of dollars. Suhanic (2001), also stated that the state of California in the 1990’s had its own share of software project failures in its bid to implement the vehicle registration and driver’s license database and the state welfare system. According to Suhanic (2001), a study of 100 failed information technology projects by KPMG International of the Netherlands showed cumulative budget of $240 million. Their final cost was $360 million. A 1995 study showed that 31 percent of software projects would be canceled. Of those completed, 53 percent will cost 189 percent of their original estimates, (Suhanic, 2001).

Although there are many tools and techniques available to assist in creating software project cost estimates, they are still very inaccurate. Several authors in software development have given varied suggestions for these inaccuracies and ways to overcome some of them. Business organizations in Nigerian in recent times have adopted Information Technology to re-engineer their business processes. This has placed demand on locally developed software that will conform to our local business procedures. In the 1980s and early 1990s, several software developing companies sprang up, but only a few is left today. Their exit from
business may be attributed to Product over price or project failures and all these are consequences of poor cost estimation.

There are fears in many quarters that if software projects continue to fail in Nigeria, it will pose a threat to our economic development because of their effects on project stakeholders. For sponsors of such software projects, it could mean loss of capital. For employees, it could mean loss of jobs, which eventually becomes societal problem; and for the contractors it could be series of law suits and psychological problems. The economy suffers also because it is money gone down the drain. There is also this theory of opportunity cost of the returns that was lost because the fund would have been invested preferably somewhere else.

This paper therefore examines the effects of costs variables/drivers on project costs estimation in Nigeria’s software development industry. In developing our cost model, consideration was given to such variables that are peculiar to Nigeria, as they will benefit our software project managers. In view of the above statements, this work is geared towards examining the effects of cost drivers on cost estimation of software development in Nigeria. This specifically involves examining the collective and individual effect of cost drivers on cost estimation on software development in Nigeria.

The next section presents the theoretical and conceptual framework of the paper while section three discusses the methodology used in the paper. Section four presents the results and discussion and finally, section five presents the findings and conclusion with policy recommendations.

2. THEORETICAL AND CONCEPTUAL FRAMEWORK

2.1 SOFTWARE PROJECT MANAGEMENT AND COST ESTIMATION

Many people and organizations today have renewed interest in project management. Until the 1980s, project management primarily focused on providing schedule and resource data to top management in the military and construction industries. This interest can be attributed to
the advent and rapid growth of the microcomputer coupled with the failure of traditional management style to address certain management problems. People in almost every industry manage projects, making project management very vital in almost every organization. Like every other discipline, software project is an endeavor that has a beginning and an end with important variables affecting its implementation as time, scope, cost, quality, etc. Over the years, the software industry has had a poor record of time and cost implementation. This mainly has been attributed to the volatility of the industry and poor cost management. This volatility has frustrated several researchers and their proposed models.

SOFTWARE PROJECT COST ESTIMATION

Project estimation is a prediction based upon probabilistic assessment. In software engineering; we predict Cost or Effort. Therefore, in project estimation, we try to predict the amount of effort and cost that are involved in implementing a project, based upon probabilistic assessment. Applying it to software projects, we now can comfortably say that software cost estimation is a process of predicting the amount of money and effort required to build a software system.

Software today has been considered the most expensive element of virtually all computer-based systems. For complex, software systems, a large cost estimation error can make the difference between profit and loss.

SOFTWARE COST MODELING TOOLS

Significant research on software cost modeling began in the early 1960s (Nelson, 1966). This led to some useful partial models in the late 1960s and early 1970s. The late 1970s produced numerous software cost modeling techniques. Among them are: Software life-cycle model (SLIM), Putnam and Myers, (1992), Checkpoint by Jones (1997), Cost Constructive Model (COCOMO) by Boehm (1981). Although most of these
researchers started working on developing models of cost estimation at about the same time, they all faced the same dilemma as software grew in size and importance it also grew in complexity, making it very difficult to accurately predict the cost of software development. This dynamic field of software estimation sustained the interests of these researchers who succeeded in setting the stepping-stone of software engineering cost models.

Like in any other field of study, the field of software engineering cost models has had its own pitfalls. The fast changing nature of software development has made it very difficult to develop parametric models that yield high accuracy for software development in all domains. Software development costs continue to increase and practitioners continually express their concern over their inability to accurately predict the cost involved. One of the most important objectives of the software engineering community has been the development of useful models that constructively explain the development life cycle and accurately predict the cost of developing a software product. To this end, many software estimation models have evolved in the last two decades based on the pioneering efforts of the above-mentioned researchers.

RELEVANT MODELS, TECHNIQUES AND THEORIES

The most commonly used model techniques include Classical multiple regression approaches, Model-based, Learning-oriented, Dynamic-based, Expert-based, and composite model. A few other models have been developed over the decades and many of them are proprietary models and cannot be compared or contrasted in terms of model structure.
Table 2.1 shows the various techniques.

<table>
<thead>
<tr>
<th>Model-Based</th>
<th>Expertise-Based</th>
<th>Learning-Based</th>
<th>Dynamic-Based</th>
<th>Regression-Based</th>
<th>Composite</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLIM</td>
<td>DELPHI</td>
<td>NEURAL</td>
<td>ROBUST</td>
<td>BAYESIAN</td>
<td></td>
</tr>
<tr>
<td>COCOMO</td>
<td>RULE-BASED</td>
<td>CASE-BASED</td>
<td>ODINARY LEAST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHECKPOINT</td>
<td></td>
<td></td>
<td>SQUARE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEER</td>
<td></td>
<td></td>
<td>COCOMO II</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Software Cost Estimation Techniques - Source: Albrecht A. (1979)

MODEL-BASED TECHNIQUE

Most of the work in model-based technique has focused on algorithmic cost modeling. In this technique, costs are analyzed using mathematical formulas linking costs to inputs with metrics to produce an estimated output. The formulae used in a formal model arise from analysis of historical data. The accuracy of the model is often improved by calibrating the model to one’s specific development environment, which basically involves adjusting the weightings of the metrics.

Kermerer, (1993) conducted a study indicating that estimates varied from as much as 85 – 610% between predicted and actual values.

Boehm’s (Boehm, 1981) Constructive Cost Model (COCOMO) is one of the model-based software cost models used commercially today. The first version of the model was delivered in 1981 and COCOMO II is now available.

COCOMO is a model designed by Barry Boehm to give an estimate of the number of man-months it will require to develop a software product. COCOMO consists of a hierarchy of three increasingly detailed accurate forms. The basic COCOMO consists of a single valued model that computes software development effort and cost as a function of program size.
expressed in estimated lines of code. This model can be applied to three classes of software projects.

Organic projects – are relatively small, simple software projects in which small teams with good application experience work to a set of less than rigid requirements.

Semi-detached projects – are intermediate software projects in which teams with mixed experience levels and less rigid requirements.

Embedded projects – are software projects that must be developed within a set of tight hardware, software and operational constraints.

The basic COCOMO computes effort as a function of program size, (Pressman, 1997).

**Basic COCOMO** equation is given as:

\[ E = a(KLOC)^b \]  
\[ D = c(E)^d \]  
\[ P = \frac{E}{D} \]

Where

- \( E \) = Effort applied in person-months
- \( D \) = Development time in chronological months
- \( KLOC \) is the estimated number of delivered lines of code for the project (expressed in thousands).
- \( P \) = number of people required

Coefficients \( a, b, c, \) and \( d \) are values that depend on the class of software project as shown in Table 2.2, Boehm (1981).

Basic COCOMO is rough order estimate because it does not recognize the differences in hardware constraints, personal qualities and experience and other things that affect cost.
Intermediate COCOMO is an extension of Basic COCOMO, and it recognizes a set of ‘cost attributes’ such as: hardware, personnel, project attributes, etc. 

The intermediate COCOMO formula is given as:

$$E = a_i(KLOC)^{b_i}EAF \quad \text{Equation 2.2}$$

WHERE $E =$ effort applied in person-months.

$KLOC =$ estimated number of thousands of delivered lines of code for the project and

$EAF =$ Effort Adjusted Factors (factors calculated from the attributes).

The coefficients $a$ and exponent $b$ are read from table 2.3

Table 2.3 Effort parameters for three modes of intermediate COCOMO

<table>
<thead>
<tr>
<th>Mode</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic</td>
<td>3.2</td>
<td>1.05</td>
</tr>
<tr>
<td>Semi-detached</td>
<td>3.0</td>
<td>1.12</td>
</tr>
<tr>
<td>Embedded</td>
<td>2.8</td>
<td>1.20</td>
</tr>
</tbody>
</table>

Source: Boehm (1981)
- Personnel attributes
- Project attributes

**TABLE 2.4: SOFTWARE DEVELOPMENT EFFORT MULTIPLIER FOR INTERMEDIATE COCOMO**

<table>
<thead>
<tr>
<th>Cost Driver</th>
<th>Description</th>
<th>Rating</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Very Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Product</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RELY</td>
<td>Requirement software reliability</td>
<td>0.75</td>
<td>0.88</td>
<td>1.00</td>
<td>1.15</td>
<td>1.40</td>
</tr>
<tr>
<td>DATA</td>
<td>Database Size</td>
<td>0.94</td>
<td>1.00</td>
<td>1.08</td>
<td>1.16</td>
<td></td>
</tr>
<tr>
<td>CPLX</td>
<td>Product complexity</td>
<td>0.70</td>
<td>0.85</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
</tr>
<tr>
<td><strong>Computer</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>Execution time constraints</td>
<td>1.00</td>
<td>1.11</td>
<td>1.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STOR</td>
<td>Main storage</td>
<td>1.00</td>
<td>1.06</td>
<td>1.21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIRT</td>
<td>Virtual machine volatility</td>
<td>0.87</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
<td></td>
</tr>
<tr>
<td>TURN</td>
<td>Computer turnaround time</td>
<td>0.87</td>
<td>1.00</td>
<td>1.07</td>
<td>1.15</td>
<td></td>
</tr>
<tr>
<td><strong>Personnel</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACAP</td>
<td>Analyst capability</td>
<td>1.46</td>
<td>1.19</td>
<td>1.00</td>
<td>0.86</td>
<td>0.71</td>
</tr>
<tr>
<td>AEXP</td>
<td>Application experience</td>
<td>1.29</td>
<td>1.13</td>
<td>1.00</td>
<td>0.91</td>
<td>0.82</td>
</tr>
<tr>
<td>PCAP</td>
<td>Programmer capability</td>
<td>1.42</td>
<td>1.17</td>
<td>1.00</td>
<td>0.86</td>
<td>0.70</td>
</tr>
<tr>
<td>VEXP</td>
<td>Virtual machine experience</td>
<td>1.21</td>
<td>1.10</td>
<td>1.00</td>
<td>0.90</td>
<td></td>
</tr>
<tr>
<td>LEXP</td>
<td>Language experience</td>
<td>1.14</td>
<td>1.07</td>
<td>1.00</td>
<td>0.95</td>
<td></td>
</tr>
</tbody>
</table>
INTERMEDIATE COCOMO SOFTWARE DEVELOPMENT EFFORT ESTIMATION

Nominal Scaling Equation

The nominal scaling equation uses the same form as those applied in Normal Basic COCOMO equation. This nominal estimate is the adjusted to applying effort multipliers determined from the project’s rating with respect to the 15 cost drivers attributes.

The equations below are derived from table 2.3

Organic Mode  
Effort = 3.2 (KSLOC)1.05

Semidetached  
Effort = 3.0 (KSLOC)1.12

Embedded  
Effort = 2.8 (KSLOC)1.20

Software Development Effort Multiplier:

Table 2.4 represents the Intermediate COCOMO software development effort multipliers. Each cost driver attributes has a set of multipliers that are keyed to a set of project rating for the attributes. These multipliers are rated from very low to extra high. The ratings reflect the expectation of the attributes on the project. These multipliers are the basic difference between Boehm’s estimation model and other models that uses lines of code for their estimation. The multipliers recognize the influence of the 15 cost drivers on software estimation cost.

Detailed COCOMO is defined in Barry Boehm’s book ‘Software Engineering Economics in 1981 [BOEH81]’ to incorporate all characteristics of the Intermediate COCOMO version with an assessment of the cost driver’s impact on each step (analysis, design, etc) of the software engineering process. Advanced COCOMO model computes effort as a function of program size and a set of cost drivers weighed according to each phase of the software lifecycle. The Advanced model applies the Intermediate model at a component level, and then a phased-based approach is used to consolidate the estimate.

<table>
<thead>
<tr>
<th>Project</th>
<th>Effort Multipliers</th>
<th>1.24</th>
<th>1.10</th>
<th>1.00</th>
<th>0.91</th>
<th>0.82</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODP Modern</td>
<td>programming</td>
<td>1.24</td>
<td>1.10</td>
<td>1.00</td>
<td>0.91</td>
<td>0.82</td>
</tr>
<tr>
<td>TOOLS Software tools</td>
<td></td>
<td>1.24</td>
<td>1.10</td>
<td>1.00</td>
<td>0.91</td>
<td>0.83</td>
</tr>
<tr>
<td>SCED Development schedule</td>
<td></td>
<td>1.23</td>
<td>1.08</td>
<td>1.00</td>
<td>1.04</td>
<td>1.1</td>
</tr>
</tbody>
</table>

Source: Boehm 1981
(Fenton, 1997). The four phases used in the detailed COCOMO model are: Requirement Planning and product Design (RPD), detailed design (DD), code and unit test (CUT), and integration and test (IT). Each cost driver is broken down by phase as shown in table 2.5.

Table 2.5: Analyst capability (ACAP) effort multiplier for Detailed COCOMO.

<table>
<thead>
<tr>
<th>Rating</th>
<th>RPD</th>
<th>DD</th>
<th>CUT</th>
<th>IT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Low</td>
<td>1.80</td>
<td>1.35</td>
<td>1.35</td>
<td>1.50</td>
</tr>
<tr>
<td>Low</td>
<td>0.85</td>
<td>0.85</td>
<td>0.85</td>
<td>1.20</td>
</tr>
<tr>
<td>Nominal</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>High</td>
<td>0.75</td>
<td>0.90</td>
<td>0.90</td>
<td>0.85</td>
</tr>
<tr>
<td>Very</td>
<td>0.55</td>
<td>0.75</td>
<td>0.75</td>
<td>0.70</td>
</tr>
</tbody>
</table>

**EXPERTISE–BASED ESTIMATION TECHNIQUES**

Expertise-based techniques are useful in the absence of quantified, empirical data. They capture the knowledge and experience of practitioners seasoned within a domain of interest, providing estimates based upon a synthesis of the known outcomes of all the past projects to which the expert is privy or in which he or she participated. The obvious drawback to this method is that an estimate is only as good as the expert’s opinion, and there is no way usually to test that opinion until it is too late to correct the damage if that opinion proves wrong. Years of experience do not necessarily translate into high levels of competency. Moreover, even the most highly competent of individuals will sometimes simply guess wrong. Two techniques have been developed which capture expert judgment but that also take steps to mitigate the possibility that the judgment of any one expert will be off. These are the Delphi technique and the Work Breakdown Structure.

**LEARNING-ORIENTED TECHNIQUES**

Learning-oriented techniques include both some of the oldest as well as newest techniques applied to estimation activities. The former are represented by case studies, among the most traditional of “manual” techniques; the latter are represented by neural networks, which attempt to automate improvements in the estimation process by building models that “learn” from previous experience.
DYNAMIC-BASED TECHNIQUE.

Dynamics-based techniques explicitly acknowledge that software project effort or cost factors change over the duration of the system development; that is, they are dynamic rather than static over time. This is a significant departure from the other techniques highlighted in this paper, which tend to rely on static models and predictions based upon snapshots of a development situation at a particular moment in time. However, factors like deadlines, staffing levels, design requirements, training needs, budget, etc., all fluctuate over course of development and cause corresponding fluctuations in the productivity of project personnel. This in turn has consequences for the likelihood of a project coming in on schedule and within budget. The prominent dynamic techniques are based upon the system dynamic approach to modeling originated by Jay Forrester nearly forty years ago, Forrester (1961).

Regression-Based techniques

Regression-based techniques are the most popular ways of building models. These techniques are used in conjunction with model-based techniques and include “Standard” regression, “Robust” regression, etc.

“Standard” Regression-Ordinary Least Square (OLS) method

“Standard” regression refers to the classical statistical approach of general linear regression modeling using least squares. It is available as an option in several commercial statistical packages such as Minitab, SPlus, SPSS, etc.

\[ y_t = \beta_1 + \beta_2 x_{12} + \ldots + \beta_k x_{ik} + e_t \]  

Where \( x_{12} \) … \( x_{th} \) are predictor (or regressor) variable for the \( t_{th} \) observation, \( \beta_2 \ldots \beta_k \) are response coefficients, \( \beta_1 \) is an intercept parameter and \( y_t \) is the response variable for the \( t_{th} \) observation. The error term, \( e_t \) is a random variable with a probability distribution (typically normal). The OLS method operates by estimating the response coefficients and the intercept parameter by minimizing the least square error term \( r_i^2 \) where \( r_i \) is the difference between the observed response and the model predicted response for the \( i_{th} \) observation. Thus all observations have an equivalent influence on the model equation.
Composite Techniques

As discussed above there are many pros and cons of using each of the existing technique for cost estimation. Composite techniques incorporate a combination of two or more techniques to formulate the most appropriate functional form of estimation.

2.3.6.2 COCOMO II

Research on COCOMO II started in the late 1990’s because COCOMO’81 is not enough to apply to newer software development practices. COCOMO II has three different models:

The application composition model, which is suitable for projects, built with modern GUI-builder (Graphical User Interface). The early design model is used to make rough estimates of a project entire cost and duration. It defines only seven cost drivers. The post-architecture model is used after projects overall architecture is developed. It has new cost drivers, new line counting rules, and new equations.

In general, COCOMO II estimates cost, derived directly from person-months effort, by assuming the cost is basically dependent on total physical size of all project files, expressed in thousands single lines of code (KSLOC).

Other factors contributing to costs and duration are the five scale drivers (precededness, development flexibility, architecture, team cohesion and process maturity); the seventeen cost drivers.

The estimation formula is:

Effort (in person-month) = a x EAF x (KSLOC)^b ........................................... 2.9

Where coefficient a is the project specified is about 3 (2.94) and is adjusted using EAF, and scaling factor b is exponent derived from the five scale drivers and is close to 1 (1.0997).

EAF = Effort adjustment factor which is a product of the ratings of the various seventeen cost drivers.

Schedule equation:

Duration = 3.67 * (Effort)SE ................................................................. 2.10

Where Effort is effort from COCOMO II effort equation.

SE is the schedule equation exponent derived from the five scale drivers.

Average staffing = Effort/Development time.
COCOMO II supersedes earlier version COCOMO such as COCOMO’81. The model specifies inception phase cost estimates by reducing the total number of parameters to seven from fifteen and introduces five new scale drivers.

This section has presented various COCOMO estimation models currently available today. The advantages of each model depend on the magnitude of the project and how precise the result of the project is intended. The key lesson to learn here is that even with the latest COCOMO there is still room for improvement.

### Table 2.7 COCOMO II Cost Drivers

<table>
<thead>
<tr>
<th>Cost driver</th>
<th>Description</th>
<th>Very low</th>
<th>Low</th>
<th>Normal</th>
<th>High</th>
<th>Very high</th>
<th>Extra high</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RELY</td>
<td>Required software reliability</td>
<td>0.75</td>
<td>0.88</td>
<td>1.00</td>
<td>1.15</td>
<td>1.39</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>Database size</td>
<td>0.93</td>
<td>1.00</td>
<td>1.09</td>
<td>1.19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPLX</td>
<td>Product complexity</td>
<td>0.70</td>
<td>0.88</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
<td>1.66</td>
</tr>
<tr>
<td>REUSE</td>
<td>Required reusability</td>
<td>0.91</td>
<td>1.00</td>
<td>1.14</td>
<td>1.29</td>
<td>1.49</td>
<td></td>
</tr>
<tr>
<td>DOCU</td>
<td>Documentation</td>
<td>0.95</td>
<td>1.00</td>
<td>1.06</td>
<td>1.13</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Platform</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>Execution time constraints</td>
<td>1.00</td>
<td>1.11</td>
<td>1.31</td>
<td>1.67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STOR</td>
<td>Main memory constraints</td>
<td>1.00</td>
<td>1.06</td>
<td>1.31</td>
<td>1.57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PVOL</td>
<td>Platform volatility</td>
<td>0.87</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Personnel</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACAP</td>
<td>Analyst capability</td>
<td>1.50</td>
<td>1.22</td>
<td>1.00</td>
<td>0.83</td>
<td>0.67</td>
<td></td>
</tr>
<tr>
<td>PCAP</td>
<td>Programmer capability</td>
<td>1.37</td>
<td>1.16</td>
<td>1.00</td>
<td>0.87</td>
<td>0.74</td>
<td></td>
</tr>
<tr>
<td>PCON</td>
<td>Personnel continuity</td>
<td>1.24</td>
<td>1.10</td>
<td>1.00</td>
<td>0.92</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td>AEXP</td>
<td>Application experience</td>
<td>1.22</td>
<td>1.10</td>
<td>1.00</td>
<td>0.89</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>PEXP</td>
<td>Platform experience</td>
<td>1.25</td>
<td>1.12</td>
<td>1.00</td>
<td>0.88</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>LTEX</td>
<td>Language and tool experience</td>
<td>1.22</td>
<td>1.10</td>
<td>1.00</td>
<td>0.91</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>Description</td>
<td>1.29</td>
<td>1.10</td>
<td>1.00</td>
<td>1.00</td>
<td>0.78</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>TOOL</td>
<td>Software tools</td>
<td>1.24</td>
<td>1.12</td>
<td>1.00</td>
<td>0.86</td>
<td>0.72</td>
<td></td>
</tr>
<tr>
<td>SITE</td>
<td>Multisite development</td>
<td>1.25</td>
<td>1.10</td>
<td>1.00</td>
<td>0.92</td>
<td>0.84</td>
<td>0.78</td>
</tr>
<tr>
<td>SCED</td>
<td>Development schedule</td>
<td>1.29</td>
<td>1.10</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
</tr>
</tbody>
</table>

Source: Boehm COCOMO 1997

COST DRIVERS

For the purpose of this work, the following cost drivers have been identified:
- Software Personnel attributes
- Software project tools attributes
- Software product attributes
- Computer hardware constraints attributes
- Software Project size
- Organizational management attributes
- Human resource requirement attributes
- Environmental attributes

RESEARCH METHODOLOGY

As a field survey, this research focuses on the effects of cost drivers/variables on software development cost estimates. Data for analysis for this work is exclusively primary data as no attempt is made to include secondary data. A well-structured and standardized questionnaire on perceived factors that affect software development efforts (costs) based on Likert five-point ordinal scale was administered to software engineers, systems analysts, programmers and other stakeholders in the domain of this study. We have limited our study to Lagos, Abuja Owerri and Kano for our sample with a view to extending our findings to the entire population (Nigeria). A total of two hundred and fifty questionnaires were distributed. When these questionnaires came back, one hundred and seventy four were found usable.
Method Of Analysis

The statistical technique of Multiple Regressions was used for data analysis. This technique helps to predict one variable from other variables, as long as there exist some relationship between the variables. The variable being predicted is usually known as dependent variable (software cost estimation) because it values is dependent on the other variables variously referred to as the independent variable. In multiple regressions, the model describing the relationship between the dependent variable and independent variables is as given in equation 3.1 below.

\[ Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon_1 \]  

\[ \text{----------- Equation 3.1} \]

Y = the dependent variable

X1, X2 …………………….Xn = independent variables.

\( \beta_0 \) = a constant value of Y when all X values are 0.

\( \beta_1 + \beta_2 + \ldots + \beta_n \) = net regression coefficients. For instance, \( \beta_0 \) measures the change in X1 while holding the other variables constant.

\( \epsilon \) = independent and normally distributed random error term with mean zero.

For the purpose of this study, our dependent variable is Y = Effort/Cost estimates in man-month hours while the independent variables are as follows:

X1 = Software Personnel attributes

X2 = Software project tools attributes

X3 = Software product attributes

X4 = Computer hardware constraints attributes

X5 = Software Project size

X6 = Organizational management attributes

X7 = Human resource requirement attributes

X8 = Environmental attributes
RESULTS AND DISCUSSION

Data collected was subjected to multiple regression analysis. The output of the multiple regressions is shown in tables 4.16, 4.17 and 4.18. Using the output of the multiple regression analysis in Table 4.18, the relationship model is stated as follows:

\[ Y = 20.401 + .065X_1 + .220X_2 - .183X_3 - .053X_4 + .167X_6 + .929X_7 + .099X_8 \]

Table 4.16 shows that using the eight variables/cost drivers in determining the extent to which they affect cost estimates of software projects led to 70.6% success, while 49.9% of the variations in the ability of these variables to predict software costs are explained by joint action in the eight variables utilized. Also the possible error in the variables is 2.112. From F-ratio table we read the following df (8, 166) at one percent and five percent alpha level to obtain 2.51 and 1.94 respectively.

**Table 4.16 Model Summary**

<table>
<thead>
<tr>
<th>Model</th>
<th>R</th>
<th>R Square</th>
<th>Adjusted R square</th>
<th>Std. Error of the Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.706a</td>
<td>.499</td>
<td>.475</td>
<td>2.112</td>
</tr>
</tbody>
</table>

a. Predictors: (Constant), X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8  
b. Dependent variables (cost estimation)

**Table 4.17 ANOVA**

<table>
<thead>
<tr>
<th>Model</th>
<th>Sum of Squares</th>
<th>Df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Regression</td>
<td>736.821</td>
<td>8</td>
<td>92.103</td>
<td>20.644</td>
<td>.000a</td>
</tr>
<tr>
<td>Residual</td>
<td>740.607</td>
<td>166</td>
<td>4.461</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1477.429</td>
<td>174</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Predictors: (Constant), X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8  
b. Dependent Variable: Software Project Cost Estimation
Table 4.18  Coefficients\textsuperscript{a}

<table>
<thead>
<tr>
<th>Model</th>
<th>Unstandardized Coefficients</th>
<th>Standardized Coefficients</th>
<th>T</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Constants)</td>
<td>20.401</td>
<td>2.501</td>
<td>8.158</td>
<td>.000</td>
</tr>
<tr>
<td>X\textsubscript{1}</td>
<td>.065</td>
<td>.112</td>
<td>.042</td>
<td>.577</td>
</tr>
<tr>
<td>X\textsubscript{2}</td>
<td>.220</td>
<td>.103</td>
<td>.161</td>
<td>2.134</td>
</tr>
<tr>
<td>X\textsubscript{3}</td>
<td>-.183</td>
<td>.089</td>
<td>-.128</td>
<td>-2.064</td>
</tr>
<tr>
<td>X\textsubscript{4}</td>
<td>-.053</td>
<td>.159</td>
<td>-.026</td>
<td>-3.36</td>
</tr>
<tr>
<td>X\textsubscript{5}</td>
<td>-.037</td>
<td>.135</td>
<td>-.016</td>
<td>-2.76</td>
</tr>
<tr>
<td>X\textsubscript{6}</td>
<td>.167</td>
<td>.111</td>
<td>.098</td>
<td>1.498</td>
</tr>
<tr>
<td>X\textsubscript{7}</td>
<td>7.929</td>
<td>.110</td>
<td>.540</td>
<td>8.453</td>
</tr>
<tr>
<td>X\textsubscript{8}</td>
<td>.099</td>
<td>.066</td>
<td>-.098</td>
<td>1.499</td>
</tr>
</tbody>
</table>

\textsuperscript{a} Dependent Variable: Project Cost/Effort Estimation

DISCUSSION OF RESULTS

Results are discussed here in the context of the research questions.

Research question one: What are the variables/cost drivers affecting cost of software development in Nigeria?

These have been identified as:

- Software Personnel
- Software project tools
- Software product
- Computer hardware
- Software Project size
- Organizational management
- Human resource requirement
- Environment
Research question two: To what extent do cost divers/variables as a whole affect cost estimation in software development projects in Nigeria?

The test of hypothesis on this research question shows that using the eight cost driver to predict cost of software development in Nigeria made a positive impact. The conclusion is drawn from the result of the statistical test in which the F-calculated value of 20.644 is greater than F-tabulated 2.51 at 5% significant level. Hence the Null hypothesis is rejected.

Research question three: To what extent does each cost divers/variables affect cost estimation in software development projects in Nigeria?

If we apply decision Rule for T-test based on p-value of 0.05, then for any cost driver to be effective, its significant level must fall between 0.0 – 0.05. Based on the above, we can accept or reject Null hypothesis for the variables in table 4.18 and conclude as follows:

- X1 = Software Personnel is not significant (0.564) accept Null hypothesis.
- X2 = Software project tools is significant (0.034) reject Null hypothesis.
- X3 = Software product is significant (0.041) reject Null hypothesis.
- X4 = Computer hardware is not significant (0.737) accept Null hypothesis.
- X5 = Software Project size is not significant (0.781) accept Null hypothesis.
- X6 = Organizational management is not significant (0.131) accept Null hypothesis.
- X7 = Human resource requirements is significant (0.000) reject Null hypothesis.
- X8 = Environmental attributes is not significant (0.136) accept Null hypothesis.

Conclusion and Recommendation

Findings and conclusion

From the analysis of primary data, we found that:

- The level of relationship existing between cost estimation of software development and the eight explanatory variables (X1, X2, …, X8) is fairly strong (R value of 0.706) indicates 70.6% correlation exist between cost estimation of software development and the eight cost drivers.

- Equation 4.1 shows that 47.5% of the variation in cost estimate of software development can be explained by the variation in the eight cost drivers when all possible error in estimation is taken into consideration. Although, this result might not be impressive, yet can be explained by inaccuracies in existing models due to factors such as volatility in the industry, specific environment, etc.
Equation 4.1 is considered a significant predictor of cost estimation of software development project in Nigeria because the F-value calculated is far greater than the critical value at $F_{0.05}(8,166)$.

In testing hypothesis 3, using T-ratio test, some variables were found significant in cost estimation of software development while others were not.

Based on our findings, the following conclusions can be drawn.

- The use of the eight cost drivers to predict cost of software development in Nigeria will make significant impact on the prizing of our software.
- The use of Software Personnel; Computer hardware; Software Project size; Environmental attributes; and Organizational attributes will make negative impact on cost prediction in software development.
- The use of Project tools, Software products and Human resources requirements will make positive impact on cost prediction in software development.

**Recommendations**

On the basis of the above findings, the following recommendations are made:

1. Software estimation should be done under tight control of certain cost variables, such as Software product, Human resources requirements, project tools, etc.
2. Software cost estimators should bear in mind, that there is no perfect model yet and other unexplained variables might affect the prediction of cost.
3. Government should create a policy that will encourage the development of Information Technology. For instance, India and Singapore has a declared policy on the use of locally developed software.
References


