International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org Volume 3 Issue 6ǁ June 2014 ǁ PP.38-43

Software Project Cost Estimation: Issues, Problems and Possible Solutions

Adanma C. Eberendu

ABSTRACT : *Software project managers have expressed concern over their inability to estimate accurately the costs associated with software development. Various cost estimation methods are available for use in software development process but concern lies in selecting better software cost estimation model to arrive at accurate cost estimation. This paper gave an overview of software cost estimation and answered the following questions: (1) what have people already done in area of cost estimation; (2) what are the types of software cost estimation model available; (3) how is the contingency allowance accounted for in software projects? Since none of the existing software cost estimation model can work accurately on its own, we proposed a hybrid model to bridge the gap and arrive at accurate and acceptable estimates.*

KEYWORDS: *cost, estimation, estimation techniques, software, software metrics*

I. INTRODUCTION

Software cost estimation provides the vital link between the general concepts and techniques of economic analysis and the particular world of software engineering. Project Cost Estimation is the task of predicting the cost, effort or productivity required to complete the project. Ali Arifoglu (Arifoglu, 1993) described it as an empirical process and also included development time as one of the requirements to complete the project. One of the most important issues confronting any software project is how to accurately predict the cost of developing the software. Software cost estimation is important for making good management decisions (Simms, 2011). There is no good way to perform a software cost-benefit analysis (Clements, 2011), break-even analysis, or make-or-buy analysis without some reasonably accurate method of estimating software costs, and their sensitivity to various product, project, and environmental factors (Somerville, 2010). Cost management is perhaps one of the most difficult aspects of project management. Project managers are well practiced at acquiring resources, assigning work, tracking progress and completing projects (Vaughan, 2011). But since there's a cost associated with nearly everything their job becomes much more complex and difficult. They need to be sure the cost of the project is estimated accurately at the beginning, budgets are assigned for various parts of the project there are the tools to control costs throughout the lifecycle of the project (Simms, 2011). The software cost estimation considers the size of the project which was traditionally determined by delivered source lines of code.

This was acceptable when programmers are rated by the number of source lines code written. Technology has eased the development process, incorporating code generators and other case tools to fasten programming. Other cost estimation metrics in use are Function-related (function points) metrics based on an estimate of the functionality of the delivered software, Application (Object) points that can be used with languages such as database programming languages or scripting languages (Albrecht & Gaffney, 1983), and Use Case Point created to estimate the software size of systems that were object oriented in nature (Ashman, 2004). Somerville (Somerville, 2010) identified eight cost estimation techniques that use these metrics. These are Algorithmic cost modelling where most researchers have been basing their arguments due to its mathematical application; others are Expert Judgement, Pricing to win, Estimation by Analogy, Parkinson's Law, Macro, Bottom-up, and Parametric techniques (Somerville, 2010). It was suggested in Travailo et al (Trivailo, Sippel, & Sekercioglu, 2012) that combining these techniques produces better result. The key cost estimation methods used in software projects are concisely enumerated with the metrics in the following sections and their respective attributes, problems, and possible solutions provided. Barton (Barton, 2011) believes that the interests of the organization are best served when it develops its own estimation model. The models that will be presented provide the organization with an initial starting point, but organization-specific factors must be determined.The remainder of this paper is organized as follows: section 2 provides a brief literature review and places this paper in the context of existing research. Section 3 describes and justifies the software cost estimation metrics; section 4 looks at different types of software cost estimation techniques. Discussion and conclusion are then presented in section 5.

II. RELATED WORKS

There is a rapid growth of models in the 1970s through 1990s (Product & Process Innovation Inc, 2009). Few new models have been developed despite the increasing importance of controlling and estimating software development. Models of 1970s and 1980s are of no interest to the present practitioner due to the growing changes in software development (Heemstra, 1992). During the past 40 years, different studies comparing software cost estimation have been published (Boehm, 1981; Albrecht & Gaffney, 1983; Putnam, 1978). These early studies used data sets of various size and environment. The general results then were poor performance of the techniques when applied to other environment uncalibrated. A typical example is the work of Kemerer (Kemerer, 1987), which compared four models: Software Lifecycle Management (SLIM), Constructive Cost Model (COCOMO), ESTIMACs and FP using 15 projects. He reported an estimation error in terms of mean magnitude of relative error between 85% for FP and ESTIMACs, 601% for COCOMO and 772% for SLIM (Wu, 1997). Kemerer concluded that FP-based cost estimation method is a better approach especially at the early phase of software development. Using a combination of Kemerer data set and COCOMO, Briand et al (Briand, Emam, Surmann, Wieczorek, & Maxwell, 1999)compared company-specific data to multi-organizational data. Their result showed that performances of the cost estimation techniques considered were not significantly different except for analogy-based technique which appeared to be less accurate. Mittas and Angelis (Mittas & Angelis, 2008) included in the comparison statistical tests to incorporate parametric information and non-parametric by introducing what they called semi-parametric technique. Their result showed that semi-parametric technique provided more accurate estimate than parametric or non-parametric technique. This study makes the contribution of evaluating and comparing many of the common software cost estimation techniques that has been used in software engineering.

III. SOFTWARE COST ESTIMATION METRICS

3.1 Source Lines of Code (SLOC)

SLOC is an old cost estimation metric which gained popularity at the early period of computer technology, giving it advantage over other cost estimation metrics. Its application may follow either unit-per-price or specific analogy approaches (Briand, Emam, Surmann, Wieczorek, & Maxwell, 1999). It is a software metric used to measure the amount of code in a software project. It is used to estimate the effort required to develop a program and productivity once the software is produced. Several cost, schedule, and effort estimation models use SLOC as an input parameter, including the widely-used Constructive Cost Model **(COCOMO)** series of models (Boehm, 1981), Putnam model (Putnam, 1978), etc. SLOC is language-dependent unlike FP which is independent of programming language and other implementation variables (Touesnard, 2004). SLOC omits blank and comment lines in its definition but some authors use non-commentary source statements (NCSS) to eliminate blank and comment lines. SLOC is a bottom-up technique where estimation starts from the basic unit (module, subroutine, function, procedure, and subprogram) of the system under development, then sum them up and carry out experimental measures (Arifoglu, 1993) such as cost-per-SLOC, number of SLOC per day per programmer. Most software estimation techniques (COCOMO, PUTNAM, etc) are SLOC-based. Software engineers believe that total project cost can be easily derived from the resulting number of person-months; hence the estimation of effort is what most researchers talk about (Keil, Paulish, & Sangwan, 2006). With the advent of GUI-based languages/tools such as Visual Basic, much of development work is done by drag-and-drops and a few mouse clicks, where the programmer virtually writes no piece of code, most of the time. It is not possible to account for the code that is automatically generated in this case (Hoh, Baik, Kim, Yang, & Boehm, 2006). This difference invites huge variations in productivity and other metrics with respect to different languages, making the Lines of Code more and more irrelevant in the context of GUI-based languages/tools, which are prominent in the present software development arena.

3.2 Function Points

Function Points is one method that has been developed by Albrecht and Gaffney (Albrecht & Gaffney, 1983) in 1979 from empirical evidence to counter SLOC. Many projects were examined with respect to their different characteristics and the size of the final products was examined, and finally a model produced to fit the data. Wu (Wu, 1997) defines function points as a method of quantifying the size and complexity of a software system in terms of functions that the software delivers to the users. Erik Stensrud (Stensrud, 1998) classified software items into transactions and data entities using FP count. FP assumes there is a correlation between the technical infrastructure development effort and application development effort since adjusted FP adds a percentage for technical complexity instead of estimating this effort separately from the amount of functionality. It considers the linear combination of five basic software components and adjusting for environmental processing complexity, called technical complexity adjustment. These five software components are further weighted as simple, average and complex (Kusumoto, Imagawa, Inoue, Morimoto, Matsusita, & Tsuda, 2002) which gives it the unadjusted function point (UFP).the technical complexity adjustment is performed by calculating the technical complexity factors, (TCF). Function point of a system can be calculated using this formula: FP = UFP * (0.65 + 0.01 * TCF). From a customer view point, Function Points can be used to help specify to a vendor, the key deliverables (Vaughan, 2011), to ensure appropriate levels of functionality will be delivered, and to develop objective measures of cost-effectiveness and quality (Stensrud, 1998). They are most effectively used with fixed price contracts as a means of specifying exactly what will be delivered. From a vendor perspective, successful management of fixed price contracts depends absolutely on accurate representations of effort According to Touesnard (Touesnard, 2004), FP relies on counting functions which may differ from individual to individual depending on the person's ability. Counting FP is manual and time consuming. Organisations can overcome these problems by practicing function points analysis consistently over a period of time

3.3 Object Points (OP)

Object Points (OP) also referred to as application point in (Somerville, 2010) to avoid misunderstanding with objects in object-oriented systems is based on counts of windows, reports, 3GL modules and rules. Banker et al (Banker, Kauffman, & Kumar, 1992) proposed OP size metric as a replacement for function points (FP) to incorporate visual widgets of the fourth generation languages (Stensrud, 1998) and the integrated Computer-Aided Software Engineering tools, and to overcome the deficiencies of the traditional lines of code and function point metrics (Issa, Odeh, & Coward, 2007). Most recent software development languages are object-oriented in nature, dealing with objects and classes. When a class has multiple relationships with other classes, the complexity of that class increases. Many researchers tried to solve this problem by mapping OP to FP, but proposing new and enhanced approach for developmental size estimation based on object model would be more apt. There is a shortage of historical project data on which to base the empirical validation of the new object points based software cost estimation model. This significantly reduces the reliability of their application in organization. Function points estimation metric is widely accepted in software industry as a veto standard for developmental size estimation (Chamundeswari & Babu, 2004), therefore new models will take time to breakeven in the industry.

3.4 Use Case Points (UCP)

RoyClem (RoyClem, 2005)explains *Use Case Points* as a project estimation method that employs a project's use cases to produce an accurate estimate of a project's size and effort. It provides the ability to estimate an application's size and effort from its use cases. Based on work by Gustav Karner in 1993 when he used use case modeling to capture the business processes and requirements of a software application by analyzing the use case actors, scenarios and various technical and environmental factors and abstracts them into an equation (Koirala, 2004). The equation is composed of four variables:

- Technical Complexity Factor (TCF).
- Environment Complexity Factor (ECF).
- Unadjusted Use Case Points (UUCP).
- Productivity Factor (PF).

Each variable is defined and computed separately, using perceived values and various constants. The complete equation is:
UCP = TCP * ECF * UUCP * PF (Ashman, 2004)
For the actor to be classified, we need to know some technical details like which protocol the actor will use. So estimation can only be done by technical experts. FP is not technical dependent.

Types of Software Project Cost Estimation Models

In this section, software cost estimation techniques will be discussed and their principles described making a distinction between them.

4.1 Expert Judgement

Expert judgement involves consultation with software cost estimation experts to use their experience of the proposed project to estimate its cost (Ramesh & Karunanidhi, 2013). Liming Wu (Wu, 1997) described expert judgement as cost estimation collaborative group where anonymous participants used their experience and understanding of the project to estimate the cost. The experts bring in their experience from past projects to the proposed project. The group uses Delphi techniques during their collaboration with a facilitator providing anonymous summary of the experts' estimates at any iteration (Heemstra, 1992; Ramesh & Karunanidhi, 2013; Somerville, 2010). It does not require data, elaborate mathematical equations (Briand, Emam, Surmann, Wieczorek, & Maxwell, 1999) or expertise but relies solely on judgement, hunch, and intuition (Arifoglu, 1993). The estimates are qualitative and not objective rather subjective (Trivailo, Sippel, & Sekercioglu, 2012). Also, it is difficult for someone to reproduce and use the knowledge and experience of an expert. If expert judgement is applied wrongly, inaccurate estimation may occur resulting to cost overrun or underrun.

4.2 Pricing to win (PtW)

The software project cost is estimated to be whatever the customer has available to spend on the project. Customer's expectations are met regarding price but it is risky for the project manager because project may be underestimated. The estimator's main interest is to win the contract without considering how much it will cost to deliver. This is the strategy used by new software developing companies who are interested to break into the business, thus, as their capability and performance increase, cost estimation increases. Pernia (Pernia, 2012) advocated for PtW when adequate lead time to gather and assess information on competitors and customers are available. Heemstra (Heemstra, 1992)hardly calls it software cost estimation technique, because commercial motives play an important role in this approach. The contractor gets the contract due to underestimation but the probability of the system meeting all the requirements is minimal.

4.3 Parkinson's Law

Parkinson's Law states that work expands to fill the time available (Somerville, 2010). The cost is determined by available resources rather than by objective assessment. If the software has to be delivered in 12 months and 5 people are available, the efforts required are estimated to be 60 person-months. In situation of overestimation, the planned effort will be used completely. In The Economist (Economist, 2014), the writer claimed that this law is based upon an analysis of the factors by which that growth is controlled. This growth is controlled in software projects by available resources (manpower, money, and time) and what grows are the delivered source instructions, function points, or object points depending on the programming language used and the case tools employed. Fortunately, there is no cost overrun here but the system is usually unfinished which is not good for the project.

4.4 Estimation by Analogy (EbA)

This technique relies on extrapolation based on comparison of previous projects which are analogous or similar in nature with the new project (Trivailo, Sippel, & Sekercioglu, 2012). The project whose cost is to be estimated is characterised by a set of attributes which is stored in the database on the previous projects. (Mittas & Angelis, 2008). Expert analysts are required to search for these attributes of the previous projects that correspond to this project such as size, team experience, technology in use, application domain, or what Barry Boehm called Cost drivers in COCOMO II (Constructive Cost Model II) (Somerville, 2010) to inform their opinion (Wu, 1997). Mittas and Angelis (Mittas & Angelis, 2008)expressed EbA as a non-parametric model, though it is an empirical technique. Trivailo et al (Trivailo, Sippel, & Sekercioglu, 2012) considered this method of estimation to be subjective but they also ascertained its limitation of identifying a suitable analogue or adequate technical data (Briand, Emam, Surmann, Wieczorek, & Maxwell, 1999). Application of this model is only possible where records of similar completed project are available, but impossible if no comparable project has been tackled (Somerville, 2010). The similarities between the previous projects and the proposed project are identified and effects estimated. The problems in this method are how best to describe a project, how many projects to be used when comparing, and database storage. The estimation may be inaccurate leading to cost overrun or underrun if inexperienced analysts are used (Ramesh & Karunanidhi, 2013)

4.5 Algorithmic Cost Model

Algorithmic cost model according to researchers (Somerville, 2010; Ramesh & Karunanidhi, 2013) is developed using historical cost information that relates some software metric (usually its size) to the project cost. Project Costs based on historical data such as source lines of code (SLOC), functionality of the system, and other cost drivers e.g. design methodology are analyzed using mathematical equations linking cost or inputs with metrics to produce an estimated output. As the size of the software increases, extra costs are incurred due to communication overhead of large team, more complex configuration management, Difficult system integration, etc; thus, the larger the system, the larger the value of this exponent (Heemstra, 1992). It is able to generate repeatable estimates, modify input data, refine and customize formulas, and support sensitivity analysis (Wu, 1997). All algorithmic models suffer from the same fundamental difficulties (Somerville, 2010) that it is often difficult to estimate size at an early stage in a project when specifications are available; function-points and object points are easier to produce than estimates of SLOC. Algorithmic models deviate from the classical view of cost estimation process but only provide a metric to measure the size of the finished system (Kemerer, 1987). Poor sizing inputs and inaccurate cost driver rating will result in cost overrun or underrun (Wu, 1997).

4.6 Macro Estimation Method

Also known as top-down estimation, an overall estimation of the software project cost is split into pieces and assigned to each part of the low-level components of the project (Heemstra, 1992; Wu, 1997; Ramesh & Karunanidhi, 2013). Sommerville (Somerville, 2010) saw the macro estimation technique as starting at the system level to assess the overall functionality of the product and how this functionality is provided by interacting sub-functions. It takes into account costs system-level activities such as integration, configuration management and documentation, which other estimation methods may ignore. This method can underestimate the cost of solving difficult low-level technical problems associated with specific components such as interface to nonstandard hardware. It requires minimal project detail yet provides no detailed basis for justifying decisions or estimates.

4.7 Bottom-up Estimation

This technique requires the project team to decompose the work into small components (Wu, 1997; Ramesh & Karunanidhi, 2013). According to Heemstra (Heemstra, 1992), the cost of individual software component is estimated by the person who will develop that component. Individual estimates are then summed up to get the overall project cost. The smaller the project activities, the easier it is to estimate. This estimate is usually more accurate because the work is very small and better understood. Conversely, it is time consuming and may be impossible to decompose activities that are hard to define (Product & Process Innovation Inc, 2009). Kemerer (Kemerer, 1987) rated the accuracy of using this method at ±20%, showing that using this method, the project may overrun or underrun by 20%.

4.8 Parametric Estimating technique

An estimating technique that uses a statistical relationship between historical data and other variables (e.g., lines of code in software development) to calculate an estimate for activity parameters, such as scope, cost, budget, and duration (Briand, Emam, Surmann, Wieczorek, & Maxwell, 1999). This technique can produce higher levels of accuracy depending upon the sophistication and the underlying data built into the model (Heemstra, 1992). This method relies on the fact that the quantity of work to be performed is directly proportional to some underlying measure of the deliverable. Parametric modeling requires historical data based on similar (Mittas & Angelis, 2008) projects and reasonable measurements of the quantities or elements of work to be performed (AACE, 2008). It is suitable for conceptual or system design phase estimates, when some of the final scope and the project approach have been defined. Its accuracy is between +75% and -25% (Kemerer, 1987). The parametric estimate uses estimating rules to approximate the likely final estimate of either the entire project or several major components of a larger project; hence the complexity of developing the model requires statistical skills and historical data with a range of risks and outcomes (AACE, 2008; Trivailo, Sippel, & Sekercioglu, 2012). The parametric estimating method is substantially more accurate when using a database of many completed projects.

IV. DISCUSSION AND CONCLUSION

Each of the classical software cost estimation techniques has advantages and disadvantages. The best approach is to combine two or more techniques to estimate project cost, we recommend **hybrid estimation technique**. The lack of accurate and reliable estimation techniques combined with financial, technical, organizational, and social risks of software projects, require a frequent estimation during the development of an application and the use of more than one estimation technique. This model can be carried out in three stages. First, the estimate of the size of the product to be developed is obtained using object points because most programming languages of today are object-oriented in nature. The second stage converts this size estimates to cost estimate using at least two of the existing techniques and the most accurate estimate will be chosen at each stage. This needs to be proved in subsequent research. The accurate estimation of software development costs is a critical issue to make good management decisions and accurately determine how much effort and time a project requires both for project managers, system analyst and developers. There are many software cost estimation techniques available but none is necessarily better or worse than others, thus their strengths and weaknesses complement each other. In this paper we considered the strengths and weaknesses associated to different types of cost estimation techniques. Our result concluded that using more than one estimation technique will give accurate estimate to avoid cost overrun or underrun. Cost estimator should combine methods based on the strengths and weaknesses of the method and its appropriateness to the project. If a project is unknown, i.e. entirely new, it is advisable to use algorithmic method or function point-based method with expert judgement especially in the early phase of the software lifecycle because of its great uncertainty values of size. If the language is of the object-oriented family, object point based techniques will be more appropriate.

We finally recommend cost estimators to use different cost estimating techniques, compare the results and determine the reasons for any large variations. It is also advisable to document all assumptions made during cost estimation and maintain historical database.

REFERENCES

[1] AACE. (2008). Contingency Estimation - General Principles. *TCM Framework Section 7.6* .

[2] Albrecht, A., & Gaffney, J. (1983, Nov.). Software Function, Source Lines of Code and Developoment Effort Prediction: A software Science Validation. *IEEE Transaction on Software Engieering* , 639-648.

[3] Arifoglu, A. (1993). A Methodology for Software Cost Estimation. *Software Engineering Notes , 18* (2), 96 - 105.

[4] Ashman, R. (2004). Project Estimation: A simple use-case based model. *IT Professional , 6* (4), 40-44.

[5] Banker, R., Kauffman, R., & Kumar, R. (1992). An Empirical Test of Object-Based Output Measurement Metrics in a Computer-Aided Software Engineering (CASE) Environment. *Journal of Management Information Systems , 8* (3), 127-150.

[6] Barton, G. (2011, JUne 2). *Project costs in translation* . Retrieved February 24, 2014, from projectmanager.com.au: http://projectmanager.com.au/managing/cost/project-costs-in-translation/

[7] Boehm, B. (1981). *Software Engineering Economics.* Englewood, Cliffs, New Jersey, USA: Prentice-Hall.

[8] Briand, L. C., Emam, E. K., Surmann, D., Wieczorek, I., & Maxwell, D. K. (1999). An Assessment and Comparison of common software Cost Estimation Modelling Techniques. *ICSE* (pp. 313-322). Los Angeles, California, USA: ACM.

[9] Chamundeswari, A., & Babu, C. (2004). An Extended Function Point Approach for Size Estimation of Object-Oriented Software. *ACM Software Engineering Notes* .

[10] Clements, S. (2011, June 7). *Project cost management and the PMO* . Retrieved February 24, 2014, from projectmanager.com.au: Project cost managem http://projectmanager.com.au/managing/cost/project-cost-management-and-the-pmo/

[11] Economist. (2014). *Parkinson's Law*. Retrieved February 26, 2014, from The Economist Newspaper Limited: http://www.economist.com/node/14116121

[12] Heemstra, F. (1992). Software Cost Estimation. *Information and Software Technology , 34* (10), 627-639.

[13] Hoh, P. I., Baik, J., Kim, S., Yang, Y., & Boehm, B. (2006). A Quality-based Cost Estimation Model for the Product Line Lifecycle. *Communications of the ACM , 49* (12), 85-88.

[14] Issa, A., Odeh, M., & Coward, D. (2007). Can Function Points be Mapped to Object Points. *The International Arab Journal of Information Technology , 4* (1), 41-49.

[15] Keil, P., Paulish, J. D., & Sangwan, R. S. (2006). Cost Estimation for Global Software Development. *EDSER'06* (pp. 7-10). Shanghai, China: ACM.

[16] Kemerer, C. (1987). An Empirical Validation of Software Cost Estimation Models. *Communication of the ACM , 30* (5), 416-429.

[17] Koirala, S. (2004, December 13). *How to Prepare Quotation Using Use Case Points*. Retrieved March 17, 2014, from codeproject.com: http://www.codeproject.com/Articles/9054/

[18] Kusumoto, S., Imagawa, M., Inoue, K., Morimoto, S., Matsusita, K., & Tsuda, M. (2002). Function Point Measurement from Java Programs. *Proceedings of ACM-ICSE* .

[19] Mittas, N., & Angelis, L. (2008). Combining Regression and Estimation by Analogy in a Semi-parametric Model for Software Cost Estimation. *ACM-ESEM'08* (pp. 70-79). Kaiserslautern, Germany: ACM.

[20] Pernia, S. (2012, June 19). Pricing to Win and Deliver. The Foundation for Enterprise Development.

[21] Product & Process Innovation Inc. (2009). *Project Management Estimation Tools and Techniques*. Retrieved March 24, 2014, from Project Management Guru: www.projectmanagementguru.com/estimating.html

[22] Putnam, L. (1978). A General Empirical Solution to the Macro Software Sizing and Estimation Problem. *IEEE Transactions of Software Engineering , 4* (4), pp. 345-381.

[23] Ramesh, K., & Karunanidhi, P. (2013). Literature Survey on Algorithmic and Non-Algorithmic Models for Software Development. *International Journal of Engineering and Computer Science , 2* (3), 623-632.

[24] RoyClem. (2005, March 22). *Project Estimation with Use Case Points*. Retrieved March 17, 2014, from codeproject.com: http://www.codeproject.com/Articles/9054/

[25] Simms, J. (2011, July 7). *Reducing the cost of project rework*. Retrieved February 2014, 24, from projectmanager.com: http://www.projectmanager.com.au/managing/cost/reducing-cost-project-rework

[26] Somerville, I. (2010). *Software Engineering* (8 ed.). London: Addison-Wesley.

[27] Stensrud, E. (1998). Estimating with Enhanced Object Points vs Function Points. *Proceedings of 13th COCOMO/SCM Forum.* Los Angeles California : University of Southern California.

[28] Touesnard, B. (2004). *Software Cost Estimation: SLOC-based Models and Function Points Model.* UNB.

[29] Trivailo, O., Sippel, M., & Sekercioglu, Y. (2012). Review of Hardware Cost Estimation methods, models, and tools applied to earlyphases of space mission planning. *Progress in Aerospace Sciences* .

[30] Vaughan, M. (2011, September 16). *Managing project cost*. Retrieved February 24, 2014, from projectmanager.com.au: http://projectmanager.com.au/education/tools/managing-project-cost/

[31] Wu, L. (1997, March 4). *The Comparison of the Software Cost Estimating Methods*. Retrieved February 24, 2014, from http://www.compapp.dcu.ie/~renaat/ca421/LWu1.html