GLOBAL JOURNAL OF MATHEMATICAL SCIENCES VOL. 3, NO.2, 2004: 179 - 184
COPYRIGHT(C) BACHUDO SCIENCE CO. LTD. PRINTED IN NIGERIA. ISSN 1596-6208

ESTIMATING SOFTWARE DEVELOPMENT PROJECT SIZE USING PROBABILISTIC TECHNIQUES

B. O. OYELAMI and S. B. OYONG

Received 14 April 2004; Revision accepted 28 August, 2004).

ABSTRACT

This paper describes the quantitative process of managing the size of software development projects by Purchasers (Clients) and Vendors (Development Houses) where there are no historical databases. Probabilistic approach was used to estimate the software project size, using the data collected when we developed a tool to estimate the time to complete a new software project, and from The University of Calabar Computer Centre. The Expected size of the Tool was estimated to be 1.463 KSLOC, but when the tool was actually coded, it was found to have 1.530 KSLOC. To validate the tool further, the expected size of revenue collection program for Equity Bank Plc in Nigeria was estimated to be 17.91 KSLOC, but when the program was coded, the actual size was found to be 17.40 KSLOC. Using a two-tailed test it was found that these results were good estimates as the sizes fell within the 99.80 % confidence limit. That is:

$$P(\mu - 3\sigma_2 \leq X \leq \mu + 3\sigma_2) = 99.8\%$$

Where
X = Expected size (KSLOC)
μ = 17.91 KSLOC= Average source lines of code (mean)
σ_2 = 0.2326 KSLOC = Total standard deviation
Standard error = 3.09
P = probability
Level of significance α = 0.1

KEYWORDS: Expected size, Uncertainty, Size range, KSLOC, Vendors.

INTRODUCTION

Estimating the size of a software system is a critical development process activity. Not only does size impact on the Technical solution, it also impacts the project management solution as well as the Commercial solution to software development projects.
Technically, there has to be a clear statement of what software is available "off the shelf" and how well or otherwise it meets the requirements, and how much of the new code should be produced. Commercial consideration involves intellectual property right with respect to the amended and new software. Frequently, purchasing organizations have paid for development and expect that ownership of that part of the software now rest with them. It is, therefore, insufficient to estimate size only once, at the beginning of the project when the least is known about the project being developed, (Ross, 1999).

It is an established fact that there is a mutual relationship between the size of a software project and the development variables of Schedule (Time), Effort and software errors. This relationship is a power (exponential) function. In organizations developing and purchasing software it is necessary to estimate the expected size of the software product to be produced. Expected size, being the major driver, instills confidence in the development estimates of the product being developed, (Putnam, 1978).

However, uncertainty exists in the expected size until the product is actually coded. This uncertainty needs to be quantified in terms of size range that encompasses the lowest and highest estimates of the software to be developed. The size range is a practical way of expressing uncertainty and hence determining the development estimate risk at any given point before the software is written, (Greene, 2001).

Putnam and Myers have suggested that projects should be broken down into small segments or sub-systems and that each sub-system should not exceed three thousand effective source lines of code (KSLOC), (Putnam and Myers,1992).

B. O. OYELAMI, Mathematical Sciences Prog., School of Sciences, Abubakar Tafawa Balewa University, Bauchi, Nigeria.
S. B. OYONG, Computer Centre, University of Calabar, Calabar, Nigeria.

According to Greene, Vendors such as system houses when bidding for development work must estimate the software size and its uncertainty when preparing their development proposals. One fundamental importance of estimating software expected size is the need for effective planning by a development group, (Greene, 1996).

The purpose of this paper is to use probabilistic approach to size all software to be developed including its uncertainty in order to evaluate the development proposals made by vendors or to prepare a realistic estimate in a development group. We designed a software and coded it in C++ programming language to achieve our results. Since software development is bedeviled by poor performance due largely to wrong delivery dates, poor reliability, increased cost of production, and inability to estimate project size correctly, it becomes necessary to look for ways of improving the industry, especially how to estimate the expected size of a new project.

METHODOLOGY

To develop the software, we first of all identified and explored the problem domain which is the end product described in human (not computer programming) language. Development is the process of transforming some one's desires into a product that satisfies those desires. The product is a set of inter-related objects or entities that are viewed as a whole or designed to achieve a purpose.
Figure 1 illustrates the process of understanding the problem domain.

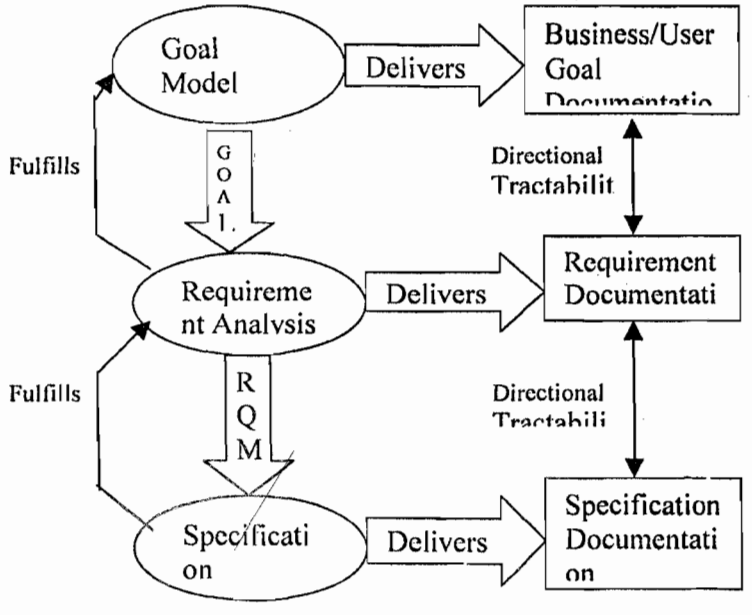

Fig. 1: Understanding the Problem Domain

The goal of this research is to improve the software project size prediction. To achieve this goal, there are requirements or constraints that must be met. These requirements are basically split into system requirements and users' requirements. In order to implement these requirements in the new system, the following specifications were considered, among others:

- Hardware specification
- Software specification
- Personnel specification
- Language type
- Application type
- Input parameters

The configuration of the computer used in implementing this research consisted of Pentium 1, 166 MHz PC with 32 MB RAM, running under Windows 98SE

In estimating the size for each sub-system, consideration was given to the following:

- Facts about the proposed software product. These facts were based on what the product was going to do when completed.
- The facts were gathered by experienced and knowledgeable people in the profession.
- Good judgement rests upon knowledge that comes from data recorded from past projects. However, these data are being made invalid by the rapidly evolving technology, as warned by Stutzke. He warned that unless an organization is at Software Engineering Institute Capability Maturity Model level 5, it may not be able to evaluate and inject new technology at the rate needed to keep up the pace. Such constant and rapid changes mean that little relevant historical data will be available to help estimate future software projects, since the half life of software engineering knowledge is typically less than three years, (Stutzke, 1986). For this reason, we relied on data gathered from probabilistic approach-estimating size in range, rather than historical data.

The size data returned for each sub-system consists of:

- Sub-system identity
- Re-used size (r)
- Minimum KSLOC size (a)
- Most likely KSLOC size (m)
- Maximum KSLOC size (b)

This size data is then summed to give the mean size of the software and the standard deviation. According to Putnam, 20 % of the mean size is computed as environmental overhead and summed up to give expected size, (Putnam, 1978), However, in implementing the technique, the authors did not consider reused software in the two case studies. Only thousands of source lines of code (KSLOC) were used.

DEVELOPMENT BASELINE

For each sub-system the skewed mean value (x) was calculated using the formulae:

$$\text{Mean Size } (X) = (a + 4m + b) / 6 \quad \ldots\ldots (1)$$

Where 6 represent the number of parameters involved (four most likely values, one low value and one high value). Viewing a 3-point estimate as defining some associated normal distribution, the High minus the Low represents a six-standard-deviation range. Dividing it by six provides an estimate of one standard deviation:

$$\text{Standard deviation } (\sigma) = (b - a) / 6 \quad \ldots\ldots (2)$$

The overall total mean size is then calculated together with the corresponding standard deviation by summing over all the sub-systems using the formulae:

$$\text{Mean size } (\mu) = \sum x_i \quad \ldots\ldots (3)$$

$$\text{Total Std dev. } (\sigma_1) = \sqrt{(\sum(\sigma)^2)} \quad \ldots\ldots (4)$$

This is used to compute the 99.8 % confidence size at $\alpha = 0.1$ level of significance, such that when the system is completed it will fall within the estimated range using the formula below:

$$\text{99.8 \% size range} = \mu \pm (3\sigma_1) \quad \ldots\ldots (5)$$

(Putnam and Myers, 1992).

TABLE 1: PROJECT SIZE DECOMPOSITION

PROGRAM MODULES	MIN	MOST	MAX	EXPTD	STD DEV	VARIANCE
Security Interface	13	25	30	23.8	2.83	8.01
Constructor ( )	5	10	17	10.33	2	4
Access function ( )	3	10	15	9.7	2	4
Destructor ( )	3	5	15	6.33	2	4

The baseline size is used to quantify and negotiate any additional requirement changes while development is in progress. Requirement changes unavoidably occur after awarding development contract. Table 1 below illustrates a sample data segment used in estimating the expected value or baseline value. For detail explanation see Oyong, (Oyong, 2003).

CASE STUDY: UNIVERSITY OF CALABAR COMPUTER CENTRE

The Computer Centre is a utility department of the University of Calabar, Calabar. It provides computer literacy training to the University community and environs. It also has a software development unit where software is developed, especially for the University consumption. The unit is now reaching out to the public for software development and has a crop of Computer science graduates as developers.

It is one of the projects developed by this unit that was used as a case study in this research work: A program to compute revenue collection for Equity Bank Plc on behalf of Custom and Excise.

RESULTS AND DISCUSSION

The result obtained from the simulation was summarized as shown below:

Expected size	1.463 KSLOC
Actual size	1.530 KSLOC
Standard Deviation	0.0267 KSLOC
99.8 % Min Size Range	1.382 KSLOC
99.8 % Max. Size Range	1.543 KSLOC
Number of modules	67
Level of significance, α	0.1

From the results above, the total mean size was found to be 1.463 KSLOC. This means that one thousand four hundred and sixty three lines of code were required (1463 SLOC) for the implementation of the project. Because this is a sample estimate of the actual population figure (which is to be realized after the program is written), the program generated a total standard deviation of 0.0267 KSLOC. This figure is a measure of how much deviation can be expected in the computation of the final number of lines of code needed for the implementation of the program.

It was also observed that the 99.8 % confidence limit has a minimum size range of one thousand, three hundred and eighty two (1.382 KSLOC) required for the implementation, and a maximum size range of one thousand five hundred and forty three (1.543 KSLOC) source lines of code that are required for the same purpose. This gives a 99.8% probability that the proportion of size estimate is equal to 1.463 ± (0.0267 x 3). The number 3 is a measure of the standard error required to ensure that the computed figure of 1.463 will fall within the range, and that it is 99.8 % sure that the size range will fit. It actually fitted as the actual thousands of source lines of code, 1.530 KSLOC, were within the maximum expected range.

The above computation was derived from the model below

$$P(\mu - 3\sigma_1 \leq X \leq \mu + 3\sigma_1) = 99.8\% \quad \text{.....(6)}$$

$$P(1.382 \leq 1.463 \leq 1.543) = 99.8\%$$

Where:

The standard error = 3.09

The standard deviation: σ_1 = 0.0267 KSLOC

Average source lines of code (Mean): μ = 1.463 KSLOC

P represents Probability

Level of significance: α = 0.1

The estimate is normal because the baseline size is large, that is, greater than or equal to 30 SLOC. This result simply suggests that any program of the given type, business, that does the same type of job

must have the actual thousands of source lines of code (KSLOC) lie within this range. This suggestion corroborates the implied null hypothesis, which states that:

H_0: The lines of code will fall within the given range

H_1: The lines of code will not fall within the given range

Since the actual thousands of source lines of code (KSLOC) produced fell within the estimated range then H_0 is accepted at $\alpha = 0.1$ level of significance. This implies that the estimated source lines of code are 99.8 % accurate. If in practice the size of the program turns out to be far larger than the estimated size, a lot of things would go wrong.

- The number of men and women put on the job would have been grossly under estimated.
- The time of completion would not be met, as the computed schedule would be inadequate.
- The project cost would face the rigor of upward review or be abandoned.
- The customer will not market the product as promised and may sue the developing company for breach of contract.

If, however, the size of the project turns out as estimated or even less, the developing company would create goodwill by delivering on time and within budget.

CASE STUDY: UNIVERSITY OF CALABAR COMPUTER CENTRE

To further test the efficiency of the tool, the size in KSLOC of a life project was simulated and the following data were arrived at

Expected size	17.91 KSLOC
Actual Size	17.40 KSLOC
Total Std Deviation, σ_2	0.2326 KSLOC
99.8 % Min size range	17.21 KSLOC
99.8 % Max. Size range	18.61 KSLOC
Total Module count	41

From the result above, the project had 17.4 KSLOC actually produced when the program was completed as against the computed value of 17.91 KSLOC. This was a good estimate as the size fell within the 99.8 % confidence limit. That is, given the range:

$$17.21 \leq 17.91 \leq 18.61) = 99.8\ \%$$

as illustrated in figure 2 .

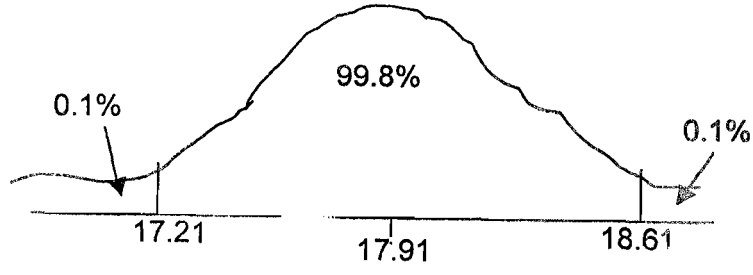

FIG. 2 Two- tailed test

We observed that the actual value of 17.4 KSLOC is greater than the minimum value of 17.21 KSLOC but less than the baseline value of 17.91 KSLOC.

CONCLUSION

In practice, size estimation and hence time and effort estimations are done either from experience (The Delphi method) or by analogy (Expert judgement), and by heavy reliance on software databanks of

past projects. In the paper we used probabilistic models to determine the baseline size of a project using data obtained from The University of Calabar Computer Centre (Vendor) to estimate the size of revenue collection program for Equity Bank Plc (Client). This research opens a gateway for software developers to obtain the baseline size of a new project before coding. There are many lessons to learn from this research work. It was seen that estimates remain estimates, but the greater fear is that technology is evolving so rapidly that stability and consistency are difficult to keep pace with. This seems to threaten the SEI –CMM measures and have rendered the use of historical databases invalid. The half-life of technological development is now less than three years, a period too short for meaningful projects to accumulate data.

REFERENCES

Greene, J. W. E., 1996. Measures for Excellence: Sizing and Controlling Incremental Software Development. QSM Ltd. Brook Green, London W14 OJL.

Greene, J. W. E., 2001. Measures for Excellence: Estimating Software Size and Uncertainty. QSM Ltd. London W14 OHP.

Oyong, S. B., 2003. Time Estimation Tool for Software Development Projects. Case Study: University of Calabar Computer Centre (Software Development Unit). Unpublished M. Sc. Thesis, Abubakar Tafawa Balewa University, Bauchi, Nigeria. 53 – 62 pp.

Putnam, L. H., 1978. A General Empirical Solution to the Macro Software Sizing and Estimation Problems. IEEE Transactions on Software Engineering, Vol. SE-4, No. 4, Arlington, VA

Putnam, L. H. and Myers, W., 1992. Measures for Excellence: Reliable Software on Time, within Budget. Englewood Cliffs, NJ: Youdon press

Ross, M., 1999. Size Does Matter: Continuous Size Estimating and Tracking. QSM Ltd. Glendale, AZ 85302. http://www.qsm.com/qw99cse.pdf

Stutzke, R. D., 1986. Software Engineering Technology: A Survey. Cross Talk Vol. 9. No. 5. 17 – 22