IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4414

Applying Erlang Distribution For Software Size Estimation

Mr. Derek F. W. Cheung
City University of Hong Kong, Derek.Cheung@cityu.edu.hk

Ho-Leung Tsoi
Caritas Francis Hsu College, Hltsoi@yahoo.com

Division of Computer Studies, City University of Hong Kong
Tat Chee Avenue, Kowloon Tong, Hong Kong - Sar, China
Ph: (852) 27888669, Fax: (852) 27888456

ABSTRACT

The Program Evaluation and Review Technique (PERT) model is one of the most popular methods for estimating software size and, sometimes, development effort. It mainly relies on expert judgment to estimate the ultimate size of a project. However, the basic assumption of this technique may be incorrect. The issue addressed in this paper concerns the weakness of the PERT model and proposes applying Erlang distribution to tackle this problem. An example has been included to show the effectiveness of this new model.
Keywords: Software Project Management, Size Estimation, Erlang distribution

1. INTRODUCTION

The major problem faced by software project developers is the prediction of the required resource level for development. It emphasizes estimation of the size of the system to be delivered in order that appropriate budgets and schedules can be agreed upon. Supposing the size of a software system and its development effort are underestimated, it means that the budgets and time are not sufficient to cover all processes that need to be developed. As a result, underestimated projects that do reach completion are often released prematurely in order to meet the budget, but these projects may omit some important features or system testing and result in incomplete and unreliable systems. Jones (1991) claims that up to 15 % of new development projects are abandoned midstream, largely due to cost overruns. It is obvious that, without accurate estimates, the planning process is likely to be unreliable and result in budget and schedule overruns.

To tackle the problem, a number of software size estimation models have been developed in the past two decades (Albrecht 1983; Boehm 1981; Putman 1978). Among them, the PERT model is a widely cited model used in software project management. The PERT model, based on the *Beta distribution*, was developed by the U.S. Navy in the 1950s to control the development of the Polaris submarine missile program (Pressman 1992). Basically, it relies on expert judgment to estimate the ultimate size of a project. To offset the bias due to psychological and personal factors, it requires three estimates, namely Most Likely, Optimistic, and Pessimistic, to form a single expected value.

Although the PERT model is widely used, the assumption of this model may be incorrect. In the PERT model, user requirements and systems specifications are needed to estimate the software size. However, they may be very unstable in the early stage of the system life cycle. The estimation process is hindered by the imprecisions of assessment. Even small changes in these factors, such as resource availability, may heavily impact on the size of the final project. Since it is impossible to remove all types of imprecision and vagueness of system specifications in the sizing process, the most effective way is to indicate the probability of error in the estimates. Unfortunately, an extra error violates the basic assumption of the PERT model. In this paper, several factors that cause imprecision will be discussed in detail. Moreover, it proposes to apply *Erlang distribution* to tackle the assumption problem in the PERT model. The last section of this paper reports on the results of a real life application that applies Erlang distribution for doing the estimation. The result implies that the proposed model improves the way of estimation.

1.1 Imprecision of Software Size Estimation

It is commonly recognized that size estimation needs to be taken as early as possible in the software development life cycle in order to cause a real impact on the development process. Unfortunately, there are a number of factors that cause imprecision in software size estimation, especially in the early stages of the software development life cycle. As a consequence, software size estimation has proved to be a difficult task in practice. Some typical examples are described as below:

(a) Uncertainty in requirement specifications

In the early stage of software development, user requirements may not be clearly known. Requirement specifications cannot be finalized until some iterations from the analysis phase to the design phase.

(b) Rapid Changes in Information Technology

The development of software application is a dynamic environment that is characterized by many changing factors such as changes in information technology. It often leads to an improvement and will be an influence on the productivity rate. Thus the effort and time to develop software will be changed. Unfortunately, the level of impact is difficult to estimate before use.

This paper appears in *Managing Information Technology in a Global Economy,* the proceedings of the Information Resources Management Association International Conference.

(c) Difficult to quantify some intangible factors

A software system is developed by many intellectual people whose activities, such as productivity, are hard to measure and determine.

2. THE PERT ESTIMATION METHOD

In the PERT model, the estimator decomposes the development cycle into a number of phases and then statistically operates on the data to obtain estimates of the application. It enables the estimator to reduce the effect of uncertainty in estimated effort of the individual person and to obtain a better estimate of the overall value. In using this model, estimators are requested to give three estimates, namely optimistic (O_i), the most likely (M_i), and pessimistic (P_i). For instance, *Optimistic time* is the shortest possible time required for completing an activity or a sub-system. *Most likely time* is the time most frequently required for completing an activity or sub-system. *Pessimistic time* is the maximum possible time required for completing an activity or sub-system. The equation for estimating the expected value of total effort in the i^{th} subsystem, E_i, is: $E_i = \frac{O_i + 4M_i + P_i}{6}$

where O_i is the lowest possible number of i^{th} subsystem,
M_i is the most likely number of i^{th} subsystem,
P_i is the highest possible number of i^{th} subsystem,

The equation for estimating the standard deviation of the number of in the i^{th} subsystem, s_i, is:

$$\sigma_i = \frac{P_i - O_i}{6}$$

The total effort, E_t, and standard deviation for the overall system, s_t (also called *sigma*), are:

$$E_T = \sum_{i=1}^{n} E_i \qquad \text{and} \qquad \sigma_T = (\sum_{i=1}^{n} \sigma_i^2)^{1/2}$$

In above equations, the mean tells you the estimated value and the standard deviation quantifies the risk for you. The standard deviation is an indication of the spread of the curve. To be 68% confident that a project will be completed at or below a given estimated value, you must take the mean and add to the standard deviation. Other confident levels are listed as below (Arifoglu 1993):

99.8%	of being between	$E_T - 3s_T$	and	$E_T + 3s_T$
95 %	of being between	$E_T - 2s_T$	and	$E_T + 2s_T$
68%	of being between	$E_T - s_T$	and	$E_T + s_T$

2.1 Problem of Using the PERT Method

Despite the fact that the PERT model is widely used, it is not free from criticism. The assumption of Beta distribution is the major weakness to be criticized from a practical point of view. Basic property of Beta distribution is that it provides positive density only for the parameters for X-axis in an interval of finite length. If the value of X-axis is outside the interval, a negative density may appear. Negative density is impossible to be used in real life situations. For instance, there are no negative manpower or Source Lines of Code in software project development.

Experience and findings from numerous studies have shown that the development of the software project involves a lot of factors that are hard to control in this ever-changing world. Even small changes in one of these factors may cause a heavy impact on the expected value. Besides, human estimators tend to be too optimistic and the expected value will be biased toward the pessimistic estimated point (Luiz 1990). As a consequence, a general point concerning the PERT method is that there is very small probability that the actual values do not fall beyond the optimistic estimates. On the contrary, there is very high probability that the actual values fall beyond the pessimistic estimates. Although the percentage for this overrun is very low (around 10%), we need to consider this situation in estimates, especially for risky projects. Considering that Beta distribution has a precise cut-off between "The optimistic" and "The pessimistic" points, it does not handle this practical problem.

3. ERLANG DISTRIBUTION ESTIMATION METHOD

Gamma distributions (including exponential) provide a wide variety of probability models for continuous variables and are often appropriate in practice. Erlang is a special case of the gamma distribution. It was first studied by the Danish engineer Erlang in connection with the distribution of waiting times in telephone networks (Anderson 1986) and is very widely used in engineering and science disciplines.

As already stated, most people tend to give estimates that are optimistic. As a result, most size estimates will not appear as symmetries. Most, in fact, will be skewed to one side of their mean. The project manager can assume that actual activity completion times are not normally shorter than the optimistic estimate, but they ought to allow for a certain percentage of activity times to take longer than the pessimistic estimate. The author admits that the percentage may be adjusted subject to the project managers' judgments or particular organizations. The way to determine the value is not fixed, but further discussion is beyond the scope of this paper.

For using Erlang distribution in estimation, three estimates, namely optimistic time, most likely time and pessimistic time, and a percentage of error will be given by estimators. Once these parameters are given from users, the shape parameter of the Erlang distribution can be determined. Steps to derive Erlang distribution are shown as below:

Erlang Distribution: $f(x;\lambda,n) = \begin{cases} \frac{\lambda^n x^{n-1} e^{-\lambda x}}{(n-1)!} & X \geq 0 \\ 0 & X < 0 \end{cases}$ mode = n(l -1)

where n is called scale parameter and l is called shape parameter

Step 1: Devise a comprehensive questionnaire that addresses the relevant project issues

Similar to the PERT model, the practitioner relies on expert judgment to estimate the size of a project. Such estimation is based on the personal experience with past projects of similar characteristics. In order to minimize the subjectivity problem, a team of people with varying levels of expertise is formed to conduct the Erlang method. This step will consolidate the judgment of a group of professional software developers. Moreover, the project will be decomposed into three different phases; namely, design phase, coding phase, and testing phase. Estimation is made for each phase. According personal feelings, such as optimistic, most likely, and pessimistic, three estimated values will be given by the estimator. All information will be collected through questionnaires.

Step 2: Solicit opinions from experts who understand the various relevant issues.

Step 3: Apply a simple statistical counting method to obtain and feedback the result to the participants.

Various statistical data will be individually evaluated as shown below:

$E_i = \frac{T_{oi} + 4T_{mi} + T_{pi}}{6}$, $E_T = \sum_{i=1}^{3} E_i$, $E_o = \sum_{i=1}^{3} T_{oi}$, $E_m = \sum_{i=1}^{3} T_{mi}$, $E_p = \sum_{i=1}^{3} T_{pi}$ and $P_e = \frac{\sum_{i=1}^{3} P_i}{3}$

where,

Optimistic man-power :	T_{oi}	for i th phase,
Most likely man-power :	T_{mi}	for i th phase,
Pessimistic man-power :	T_{pi}	for i th phase,
Average man-power :	E_i	for i th phase
Percentage of error :	P_i	for i th phase,
Average percentage of error :	P_e	for whole system
Total man-power :	E_t	for whole system
Overall optimistic man-power :	E_o	for whole system
Mode of man-power :	E_m	for whole system
Overall pessimistic man-power :	E_p	for whole system

Step 4: Ask the participants to reassess their judgment based on the feedback information and return with any appropriate comments.

Step 5: Repeat step 3 and step 4, until no major changes are found in the Participants' opinions

The following information will be finalized after the end of the above processes:

Mode of man-power (E_m) = $n(\lambda - 1)$

$\therefore \quad \lambda = (E_m + n) / n$ ——————— (1)

Expected percentage (P_o) = $1 - P_e$ ——————— (2)

$$P_o = \int_{\min}^{\max} f(x;\lambda,n)dx = \int_{\min}^{\max} \frac{\lambda^n x^{n-1} e^{-\lambda x}}{(n-1)!} dx \quad \text{----------(3)}$$

where n is called scale parameter and l is called shape parameter.

Using Incomplete Gamma Function table (Anderson 1986), shape and scale parameters can be determined by using computer iteration method.

4. AN EXAMPLE

In this section, an example will be presented to show how to apply the Erlang estimation model for a real life software development. The example has been modified from a real-life project, which has been requested by a large jockey club and developed by an international software consultation organization. This project was designed to replace the existing telephone betting system on a proprietary hardware platform. The client request to develop this system on a Personal Digital Assistant (PDA) computer and several advance wireless transmission technologies will be included. After an intensive investigation, a decision has been reached applying Erlang distribution model to estimate the size of this application. The reasons are shown as below:

- One serious problem faced by the developer is that no sufficient historical data can be referred either from the company or public standard. It affects the accuracy in using many popular software estimation models.
- Object-Orient technology will be applied in this development and many current sizing models, such as Function Point, Feature Points, can be used in such platform.
- The application had been built on a new developed PDA platform working together with an advanced wireless digital modem. Unfortunately, the modem has not been released by the supplier while starting to develop the application.
- The development team consists of a mix of experienced software consultants and some newly recruited graduates. They do not have much experience, either in the PDA hardware platform or in wireless telecommunication technology.

In summary, there are many uncontrolled and unpredictable factors existing in the development cycle. According to the weakness of the PERT model mentioned in section I, the company derives an Erlang estimation model. It is expected to provide flexibility to the management for planning and controlling development and maintenance projects. Estimation process is shown as below:

Step 1: Devise Questionnaire to all members of the develop ment team
Step 2: Solicit opinions
Step 3: Applying statistical method and feedback result to all participants
Step 4: Re-collect opinions from participants
Step 5: Repeat step 3 and step 4 until no major changes

The total expected value X (in KSLOC-thousand source line of codes) is aggregated by all individual stages:

Optimistic size	: E_o	=	1 KSLOC
Most Likely size	: E_m	=	3 KSLOC
Pessimistic size	: E_p	=	7 KSLOC
Average Percentage of error	: P_e	=	5%

Using (1) λ = (mode + n) / n = (2 + n) / n —— (I)
Using (2) Expected percentage (P_o) = 1 - P_e = 1 - 5% = 95% —— (II)

Thus, the probability that the project finish between optimistic and pessimistic time is 95%.

Using (3) Expected percentage -95%- $\int_{\min}^{\max} f(x;\lambda,n)dx = \int_{\min}^{\max} \frac{\lambda^n x^{n-1} e^{-\lambda x}}{(n-1)!} dx$

$\int_0^{\max} \frac{\lambda^n x^{n-1} e^{-\lambda x}}{(n-1)!} dx - \int_0^{\min} \frac{\lambda^n x^{n-1} e^{-\lambda x}}{(n-1)!} dx = (\lambda^{n-1})[\int_0^{\max} \frac{x^{n-1} e^{-\lambda x}}{(n-1)!} d(\lambda x) - \int_0^{\min} \frac{x^{n-1} e^{-\lambda x}}{(n-1)!} d(\lambda x)]$

Before using computer iteration method, let y = lx, then Incomplete Gamma function can be used in above equation.

$95\% = (\lambda^{n-1})[\int_0^{\max} \frac{x^{n-1} e^{-y}}{(n-1)!} d(y) - \int_0^{\min} \frac{x^{n-1} e^{-y}}{(n-1)!} d(y)] = l^{n-1}$ [F(max,n) - F(min,n)]

Where F(x,n) is Incomplete Gamma function

By using computer iteration method, we find n = 5 and l = 1.22

The Erlang distribution is F(x:1.22,11) (see figure 1) and the expected value (50%) is 3.5 KSLOC (see figure 2). It is very close to the result (3.3 KSLOC) yielded by the PERT method.

Finally, the actual size of the project is 7.3 KSLOC. The discrepancy between the estimated and real size values for the

PERT model is +4 KSLOC (121%). The most interesting point is that the actual size of the project is greater than the pessimistic value in the PERT model. It is a good example to show that the assumption of the PERT model is incorrect in some cases, especially for some risky software projects.

5. CONCLUSION

There is no doubt that project managers have to confront many different kinds of problems, such as technical, management, and personnel, in the software development cycle. They are responsible for ensuring that all development processes are controlled appropriately in order to meet the budget and schedule. Unfortunately, software projects regularly get out of hand in the development cycle.

As mentioned earlier, software project development has a dynamic nature. Therefore, changes are bound to occur. A good estimate is the first step to get a successful software project. In this paper, a technique, applying Erlang distribution for software size estimation, has been proposed. The Erlang estimation model is not a completely new method but it tackles the weakness of using the PERT model. The author admits that this model is still subjective, since it ultimately depends on expert judgment to estimate the ultimate size of a project. However, this model incorporates confidence concept to show the risk level in practice. Instead of a crisp value, the Erlang model provides an estimate with a different confidence level that allows project managers to consider the worst situation of the development effort estimation. Besides, project managers can select appropriate confidence levels for their projects.

From the example, it shows that the pessimistic estimated value in the PERT model may not be the upper boundary of a project. Overruns are common in the software industry. As a result, the assumption of the PERT model needs to be re-considered. Besides, the Erlang approach has proved to be more informative than the PERT model. The project managers are free to select the appropriate confidence level for their project. With a higher probability level, the project management would be able to prepare a more realistic project plan to cope with the possible problems during the project development. The software project is eventually successful. It is not because there are no problems during the development cycle but because the problems are taken into account as early as possible.

ACKNOWLEDGMENT

The author is thankful to Dr. C. Yau for various insights concerning the ideas upon which this paper is built, and to Mr. Vincent Cheng and Miss Susan Addison for their helpful suggestions.

REFERENCE

Albrecht J. Allan and Gaffney E. John (1983),"Software Function, Source Lines Code, and Development Effort Prediction: A Software Science Validation," IEEE Trans. of SE, November, Vol. 9 No. 6, pp. 639-647

Anderson D.R.(1986), "Statistics: Concepts and Applications", West Publishing Company, NY.

Arifoglu A. (1993), "Methodology for Software Cost Estimation", ACM SIGSOFT.

Boehm W. Barry (1981), "Software Engineering Economics," NJ, Prentice Hall.

Pressman (1992), "Software Engineering", McGraw-Hill, NY.

Putman, L.H. (1978), "General Empirical Solution to the Macro Software Sizing and Estimating Problem," IEEE Trans. on Software Engineering, April, Vol.4 No. 4, pp.345-361

Jones, Capers (1991), "Applied Software Measurements", McGraw-Hill, NY.

Luiz A. L. (1990), "Software Size Estimation of Object-Oriented Systems", IEEE Trans. of SE., Vol. 16 No. 5, pp.510-522