Chapter 3

Applying Erlang Distribution For Software Size Estimation

Derek F. W. Cheung
City University of Hong Kong

Ho-Leung Tsoi
Caritas Francis Hsu College
City University of Hong Kong

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.

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 mid-stream, 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 chapter, 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 chapter 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.

**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.
Asymmetric Interaction in Competitive Internet Technology Diffusion: Implications for the Competition Between Local and Multinational Online Vendors
Peijian Song, Cheng Zhang, Yunjie Xu, Ling Xue, Ke Wang and Chenghong Zhang (2013). Global Diffusion and Adoption of Technologies for Knowledge and Information Sharing (pp. 221-240).
www.igi-global.com/chapter/asymmetric-interaction-competitive-internet-technology/72189?camid=4v1a