Chapter 2.4
Effort Estimation Model for each Phase of Software Development Life Cycle

Sarah Afzal Safavi
COMSATS Institute of Information Technology, Pakistan

Maqbool Uddin Shaikh
COMSATS Institute of Information Technology, Pakistan

ABSTRACT

The assessment of main risks in software development discloses that a major threat of delays are caused by poor effort / cost estimation of the project. Low / poor cost estimation is the second highest priority risk [Basit Shahzad]. This risk can affect four out of a total five phases of the software development life cycle i.e. Analysis, Design, Coding and Testing. Hence targeting this risk alone may reduce the overall risk impact of the project by fifty percent. Architectural designing of the system is a great activity which consumes most of the time in SDLC. Obviously, effort is put forth to produce the design of the system. It is evident that none of the existing estimation models try to calculate the effort put on designing of the system. Although use case estimation model uses the use case points to estimate the cost. But what is the cost of creating use cases? One reason of poor estimates produced by existing models can be negligence of design effort/cost. Therefore it shall be well estimated to prevent any cost overrun of the project. We propose a model to estimate the effort in each of these phases rather than just relying upon the cost estimation of the coding phase only. It will also ease the monitoring of project status and comparison against planned cost and actual cost incurred so far at any point of time.

DOI: 10.4018/978-1-61350-456-7.ch2.4

BACKGROUND AND MOTIVATION

Existing estimation techniques such as Functions point estimation and use case estimation rely upon the artifacts generated in earlier phase. These artifacts (i.e. Use case diagrams, class diagrams, sequence diagrams, activity diagrams, state chart diagrams etc) depict the architectural design of the entire system. These diagrams are not generated out of a blue or are not instantly available without putting any effort.

Standard task set and the percentage of work duration associated with it decomposes the ratio of effort put in each phase.

It is evident in Table 1 that although major ratio (i.e. 40%) of work effort is put in code and unit test phase. The rest 60 percent effort is put in different areas of the project development life cycle. Hence this signifies the importance of estimating cost for these phases of software development life cycle.

Usually the effort estimation is done after the analyses phase when the project reaches into coding stage. The cost / effort is measured in terms of line of codes for each functionality to be incorporated into the software. Therefore it is very clear to understand that only 40% (i.e. as shown in Table 1) of the total software development effort is estimated. Whereas this estimation is delayed until all the analyses and design has completed. We have adapted a different approach and suggest that effort estimation shall be carried out for each phase of the development process.

Table 1. Standard task set & work duration %age [4]

Activity	Standard Work Effort%
Definition Phase	
Business Requirements	6%
Functional Specifications	10%
Delivery Phase	
Detailed Design	14%
Code and Unit Test	40%
System Testing	20%
User Acceptance Testing	10%
Total Effort	**100%**

We propose this model to avoid the risk of low cost estimation as earliest as possible in the development process.

Current software cost estimation methods first try to know the size of the software to be built. Based upon this size the expected effort to be put is measured. Estimated effort further is utilized to calculate the duration (i.e. Time required) and cost (monetary/human resources) of the project.

Calculating the size of project is the foremost logical step to be taken in order to estimate the effort. If we do not know the distance to be travelled we can not estimate the cost and duration per mileage. Therefore we also first measure the size of the entire project.

We know that there are mainly three categories of software projects i.e.,

- **Organic mode:** These are relatively small, simple SW projects (application programs e.g. Thermal analysis program)
- **Embedded mode:** System programs which are developed within tight HW, SW and operational constraints (flight control SW for aircraft).
- **Semi-detached mode**: An intermediate level (size and complexity, utility programs) SW projects with mixed experience, mixed requirements. It can be mixture of organic and embedded software as well.

Therefore these categories of the software project would effect the estimation of each phase. We propose the modular approach to be adapted for the development efforts so that even large scale enterprise information systems can also be decomposed into a mix of several modules of organic, semi detached, and embedded system.