A Framework for Top-down Cost Estimation of Software Development

Tsuneo Yamaura, Tohru Kikuno
Department of Informatics and Mathematical Sciences,
Graduate School of Engineering Science, Osaka University
1-3 Machikaneyama, Toyonaka, 560-8531 Japan
yamaur_t@soft.hitachi.co.jp kikuno@ics.es.osaka-u.ac.jp

Abstract

The Function Point Method, estimation by analogy, and algorithmic modeling are three of the most commonly applied methods used to estimate the costs and worker hours needed for a software development project. These methods, however, require a deep and wide expertise in particular areas and may still result in unacceptable discrepancies between the estimated costs and the actual costs. This paper presents a framework for a top-down cost estimation method (TCE). The method is based on the assumption that different types of software have different intrinsic complexities. We expect that this method will produce easier, faster, and more accurate estimations in the early stages of a software project.

1. Introduction

Cost estimation by analogy is one of the methods most widely used to calculate the costs and worker hours required for a software development project. The basic steps are quite simple: pick a previous project that developed a similar product, estimate the software size (in KLOC) and also the worker hours, and finally calculate the cost using the heuristic .

The estimation-by-analogy approach requires experience in developing particular software (usually experience at the project manager level), and yet this approach can still produce excessive gaps between the estimated costs and the actual costs.

Bottom-up estimations are exemplified by the Function Point Method [ALB83] (hereafter called FPM). Such bottom-up estimations are supposed to be among the more promising estimation methods because FPM considers details of the functional aspects of the software, as listed below, and has a mathematical and scientific atmosphere.

Number of FPs = f4(sum of functions)

Estimated cost = f5(FPs, environment)

FPM, however, requires time to learn and also requires a deep and wide expertise in developing the target software. Even for an expert, counting function points is quite time-consuming. One of the tricky features of FPM is that the target system must be complete and should not include open ends, which means that a client/server system can be measured, but not the client part alone.

2. Basic Steps for Top-down Cost Estimation

This paper proposes TCE (Top-down Cost Estimation) that estimates the costs and worker hours required for the target software by the following basic steps:

(1) Search a software functional classification table for the same type of software with matching functions, such as a word processor, and identify the standard cost for that type of software.

(2) Adjust the standard cost by considering the developer's business strategy such as "the top priority is maintaining the shipping date" rather than "the top priority is maintaining quality."

(3) Re-adjust the above adjusted standard cost by considering the development environment (such as the ability of the programmers or the availability of hardware and software tools).

3. Assumptions in TCE

TCE has two fundamental assumptions:

(1) Each type of software has its own intrinsic characteristics: such as functional complexity, performance requirements, and sophistication level of the user interface.

(2) The software development costs and worker hours are both affected by software characteristics, corporate strategy, and the available development environment.

3.1 First Assumption: Each software program has intrinsic characteristics

The estimation-by-analogy model, the algorithmic modeling such as COCOMO, and the FP method assume that 10 KLOC developed for an online program, for example, will require the similar cost and worker hours to build the 10 KLOC of a batch program. TCE, however, assumes that each type of software has its own complexity and difficulty in design and implementation.

3.2 Second Assumption: Functions, strategy, and environment affect the cost

Three major components that affect both the software development cost and required effort are:

(1) software characteristics (e.g., functional complexity, performance requirements)

(2) corporate strategic characteristics (e.g., "ship now, repair later" or "debug now, ship later")

(3) development environment characteristics (e.g., available hardware and software tools)

The second assumption in TCE is that the standard development costs and worker hours are closely related to, and directly affected by, the software characteristics. This paper assumes that the standard costs and worker hours are 100% proportional to the software characteristics, and the estimated costs and worker hours can be calculated from the standard cost, strategy, and environment as formulated below:

- Standard cost = F1(characteristics)

0-7695-0368-3/99 $10.00 © 1999 IEEE

- Estimated cost = F2(standard cost, strategy, environment)

3.2.1 Software Characteristics

The software characteristics are absolute values associated with each type of software (i.e., software characteristics vary with the type of software), and these values are not affected by, for example, a programmer's ability, target quality level, or hardware and software configurations provided to develop the software. Software characteristics include:

(1) Functional complexity (i.e., how difficult it is to design and implement the target software).

(2) Performance requirements (e.g., the memory and disk constraints, number of concurrently operable processes, response time).

(3) Sophistication level of the user interface (e.g., line-mode interaction, full-screen GUI).

3.2.2 Corporate Strategic Characteristics

Each software development company has its own strategies when developing target software. The strategies vary the emphasis placed on any of the time of shipment, available budget, engineering (and business) decisions on the quality level, and targeted market

Of the above four, the most crucial strategy is the shipment timing. In general terms, the two contrasting strategies are:

(1) the highest priority is to ship the product as early as possible to gain market share

(2) the highest priority is high quality to get rid of the vicious circle of poorer quality software demands more worker hours and higher maintenance costs after the software is released, and the resulting lower budgets and fewer human resources lead to another poor-quality project.

3.2.3 Development Environmental Characteristics

These are the characteristics of the resources that a project can (or sometimes "must") employ to develop the target software. The development environment includes:

(1) Human resources (e.g., number of engineers assigned, ability of programmers)

(2) Hardware resources (e.g., machines, disk spaces, network bandwidth)

(3) Software resources (e.g., available tools, quality of reused software)

(4) Development strategy (e.g., waterfall model, RAD, spiral model, prototyping)

4. Building a TCE

Implementing and evaluating an operable TCE system must go through four phases. The four phases are briefly explained below, and we finish with a tentative example of how a usable standard cost table might appear.

Phase 1 (Construct a software taxonomy table)

In phase 1, we make a "software taxonomy table" that covers all software products. Of course, there are various ways to classify software. The following list could be used as the basis for developing such a taxonomy:

(1) Operating systems: job management, data management, task management, device drivers

(2) System utilities: security, file management, library management

(3) Network: Internet, client/server system, dataware, groupware, network management, distributed object environment, network protocols, infrastructures

(4) Language processors: COBOL, C/C++, FORTRAN, JAVA, documentation languages (e.g., SGML), interpreters

(5) Database: tree structure, network structure, relational database, distributed databases

(6) PC-related package software: word processor, spreadsheet

(7) Applications: banking and securities system, reservation system, financial system, inventory control system, electronic commerce,

Phase 2 (Construct a standard cost table)

In phase 2, we provide the following information for each type of software:

(a) Standard cost

(b) Weightings to correspond to emphasized goals such as "performance is not a major consideration," or "critical."

(c) Weightings to correspond to emphasized GUI goals such as "a simple GUI is enough," "a meticulously designed GUI is essential."

Phase 3 (Develop adjusting procedures)

In phase 3, we provide weightings for reflecting corporate strategic characteristics, and then provide weightings that reflect the environmental characteristics.

Phase 4 (Perform experimental evaluation of the TCE)

In phase 4, we evaluate the predictability and sensitivity of the TCE.

5. Conclusion

In this paper, we described the basic idea for a top-down method for estimating software costs. With this method we expect to be able to make relatively quick and easy estimations in the early stages of software development, without any deep or wide expertise. We described the four phases required for building an operable TCE system. We are currently implementing and analyzing a TCE prototype based on actual project data.

Bibliography

[ALB83]: Albrecht, A., Gaffney, J.Jr., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," IEEE Transactions on Software Engineering. Vol.9, No.11, pp.639-648, 1983.