Comparing the top-down and bottom-up approaches of function point analysis: a case study

CHUK YAU and LUCK-YUAN GAN

1 School of Computing and Information Technology, Griffith University, Nathan, Queensland 4111, Australia
2 Harnischfeger of Australia

Received 13 October 1993

Function point analysis is a widely cited method for estimating software project size, which is an important activity of project management. At the beginning stage of planning, the top-down approach can be applied. Having obtained more systems specifications at later stages, the bottom-up approach might also be used to improve the accuracy of the estimation. However, the bottom-up approach is not a conventional way of function point analysis. There was no empirical evidence showing the difference between the fully informed top-down approach and the bottom-up approach. Through the implementation of a function point analysis system in an in-house software development department, this paper compares the results of the two approaches. This comparison study shows that the bottom-up approach does not contribute a significant added value to a fully-informed top-down approach. Therefore, the fully-informed top-down approach has been chosen as a method for building a software metric database in the organization. More important, the observations and experience gained from this project may help in-house development organizations to establish their own function point analysis systems.

Keywords: software size estimation, function point analysis, software metric

1. Introduction

Software development organizations have struggled for years in search of an acceptable quantitative tool for managing the software costs and measuring effectiveness of the software process. The concerns of establishing a software measure include: how to measure, when to measure, what to measure and when to cease measuring and start building software. The objective of measuring software is not just to watch for trends, but to provide means whereby later software projects can be measured against, and therefore can be better managed and controlled, and thus producing better and better quality software on an on-going process.

Concerning 'how to measure', numerous software metrics have been proposed in the past. Source lines of code (SLOC) was the most popular metric employed by the software industry. SLOC has been used as the basis of software size estimation such as Putnam's SLIM model (Londeix, 1987 and Kemerer, 1987) and Barry Boehm's COCOMO model (Kemerer, 1987 and Boehm, 1981). The main disadvantage of the SLOC count is that it is not available until the coding stage of the software development life cycle has been completed. It is very difficult to express software size in terms of line of code at the early stages of software development.
Moreover, the practicality of relating SLOC to other software elements is also an important issue that puzzles software engineers (DeMarco, 1982). Another main weakness of SLOC is its language dependence. SLOC measures penalise high-level languages and often move in the wrong direction as productivity improves (Jones, 1986).

Because of the above SLOC weaknesses, function point metric can be considered as an alternative. Function point analysis was introduced by A.J. Albrecht of IBM in the middle of 1970s (Behrens, 1983). Subsequently, a variety of function point analysis methods, such as Mark II (CCTA, 1991; Symons, 1988, 1991) and the Software Productivity Research Method (Jones, 1991; Arthur, 1983), have also been reported. In general, function point is a synthetic metric that comprises the weighted totals of the inputs, outputs, enquiries, logical files and interfaces belonging to an application. Once an application's function point count is known, the metric can be used for a variety of purposes (Jones, 1991), such as software development effort estimation, productivity analysis and quality control.

Concerning 'when to measure', function point analysis can be carried out at any stage of the software process (CCTA, 1991; Symons, 1991). As shown in Fig. 1, the top-down approach can be applied at the feasibility study stage. In the top-down approach, the size of system is estimated from its functional and technical characteristics, and the environment in which the system will be built. Normally, the system characteristics and the development environment will be known early in the software development life-cycle towards the end of the feasibility study phase. As the life-cycle progresses, more information will be available and the accuracy of the top-down approach can be improved. By the time the system analysis phase has been completed, sufficient information will be available to undertake the fully informed top-down estimation.

Having obtained more systems specifications at analysis and design stages, the bottom-up approach can also be used to determine the project size. In this paper, the bottom-up approach is defined as a fully informed task-to-task function point analysis which evaluates the decomposed tasks defined in the systems design specifications. Each of the tasks is estimated, then summed to arrive at the required total effort for the activity. The bottom-up approach is not a standard method for function point analysis, but it might provide added value by improving the accuracy of size estimation. However, there was no empirical evidence showing the difference between the fully informed top-down approach and the bottom-up approach. An investigation is needed to determine the difference between the two approaches, so that users are able to select the most appropriate approach for project estimation.

There is no silver bullet for solving project sizing problems. The most important element of a good sizing system is a software metric database which allows project managers to compare the project characteristics and productivity factors of the current project with previous projects. However, the establishment of a software metric database relies on the approach for data collec-