Software Functional Complexity Analysis and Effort Prediction

De Tran-Cao[1], **Ghislain Lévesque**[2]

[1]*College of information technology, University of Cantho, Vietnam*

[2]*Computer Science Department, University of Quebec at Montreal, Canada.*

Abstract

This paper introduces a new method for analyzing and quantifying software complexity. The method is specification-based as Function Point Analysis (FPA) and its derivatives. The term "software functional complexity" refers to the cognitive difficulty derived from software functionalities. It is taken as the complexity of the task that must be fulfilled by software. The method captures complexity in input, output data, in data manipulation and in relationships between software components. These measures will be used to build a prediction model of effort needed for development or maintenance software. The prediction model is empirically established with a group of 15 software maintenance projects. The multiple regression analysis between the real effort and these measures supports the conclusion that these complexity measures can be used to predict development/maintenance effort with a fairly good precision. A cross test of the type *leave-one-out* between these 15 projects confirms also the goodness of our measures in estimating effort.

Keywords: software size, software complexity, complexity measurement, effort estimation, functional complexity measurement, task complexity.

1 Introduction

Software size is a key measure for many cost and effort estimation models. Traditionally, the number of lines of code (LOC) was often used as an intuitive measure of size. Studies [11,17] confirm that LOC is an effective measure for maintenance effort prediction. Effort estimation models such as SLIM [19] and COCOMO [6] are based on LOC. However, LOC cannot be measured early in the software development process. Furthermore, evidence suggests that LOC can be very inaccurate because they depend on language and development tools [8].

The functional approach for software sizing and for estimating development effort was proposed by Allan Albrecht in 1979 [4]. This method measures software size in terms of *function points* that is the amount of functionalities of software delivered to users. The Function Point Analysis (FPA) of Albrecht is well known because of its great advantages: independent of programming language and technology, easy to understand for client and user and applicable at early phases of the software life cycle.

The FPA's process for quantifying software size begins with identifying the elements (-type) of software from the software specifications. Then, these elements are weighted according to their complexity. Therefore, software sizing cannot be independent of software complexity measurement. Unfortunately, software complexity has not been precisely addressed in the method. The evaluation of complexity is basically subjective. It makes it difficult to be applied coherently and repeatedly. In addition, historically, FPA was developed in and designed for Management Information System (MIS). It is not appropriate to size software of other types, including real-time, scientific and embedded software. There have been many other proposals [2,14,20,22,29] that try to overcome these weaknesses. These works focus on how to reduce the