Available online at www.sciencedirect.com
SCIENCE DIRECT
Information and Software Technology 47 (2005) 881–890
INFORMATION AND SOFTWARE TECHNOLOGY
www.elsevier.com/locate/infsof

Automated software size estimation based on function points using UML models

Aleš Živkovič*, Ivan Rozman, Marjan Heričko

Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, SI-2000 Maribor, Slovenia

Received 6 December 2004; revised 21 February 2005; accepted 23 February 2005
Available online 23 May 2005

Abstract

A systematic approach to software size estimation is important for accurate project planning. In this paper, we will propose the unified mapping of UML models into function points. The mapping is formally described to enable the automation of the counting procedure. Three estimation levels are defined that correspond to the different abstraction levels of the software system. The level of abstraction influences an estimate's accuracy. Our research, based on a small data set, proved that accuracy increases with each subsequent abstraction level. Changes to the FPA complexity tables for transactional functions will also be proposed in order to better quantify the characteristics of object-oriented software.

Keywords: Function points; Software size measure; Project planning

1. Introduction

The focus of scientific research regarding object development and component-based development has already shifted from implementation to earlier development activities in software and information system development. Additionally, emphasis has also been placed on all aspects of software development that have been investigated in the context of structured techniques, from executable specifications, testing strategies to estimation models and metrics. In this paper, we will focus on one metric only: the size estimation metric for object-oriented development. Software size contains important information for project planning. Costs and schedule estimates depend on its existence and its accuracy; indirectly the project's success also depends on it. A software project is successful, if the requirements are fulfilled and no budget or deadline overflows occur [25]. With systematic size estimation, the risk of overflows is lower. Systematic software size estimation requires a method that defines a procedure for measurement, involving units and accuracy. In general, Functional Size Measurement (FSM) methods, as defined in ISO/IEC TR 14143 [12–15] can be categorized into two groups. In the first group, there are technology-independent methods; an example would be the Function Point Analysis (FPA) method [10]. In the second group, there are technology-dependent methods; for example Lines of Code (LOC) or number of classes. The methods from the first group have obvious advantages over the methods from the second group. The ultimate goal is to use only technology-independent methods. The FPA method dates back to 1979 [2] and has been updated several times. However, the core concepts and the counting procedure remains the same. Every information system processes some data that can be stored in the application database or is taken from external applications. Four operations are performed on data records: create, read, update and delete. Besides that, information systems use several query functions for data retrieval and report construction. Each record consists of several fields of basic data types or another record that can be further deconstructed. The FPA method quantifies: the number of fields in each record, the distinct operations performed on these records, and the number of these operations that are necessary to perform a business function. The sum over all business functions, multiplied with some empirically determined weights, represents the unadjusted function points value.

* Corresponding author. Tel.: +386 2 235 5115.
E-mail address: ales.zivkovic@uni-mb.si (A. Živkovič).

0950-5849/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2005.02.008

The final calculation is made using a Value Adjustment Factor (VAF) that measures system complexity. Based on the FPA method, several methods like Feature Points, Full Function Points, Function Weight, Function Bang, Mk II Function Points Analysis, COSMIC-FFP and NESMA evolved. A detailed comparison of the selected methods can be found in Ref. [33].

Although the methods are technology-independent, their use in object development is quite difficult. Methods use their own abstraction to represent a software system in a convenient way, so as to perform size count. In object development, the Unified Modeling Language (UML) is used to represent the software system as an abstraction. To overcome the gap between these two abstractions, the mapping that transforms the elements used in one abstraction to the elements of the other abstraction, has to be defined. This paper focuses on OO-to-FPA mappings. With OO-to-FPA mapping defined, the late analysis and design size estimation problem is solved, however the estimation cannot always be applied early in a software development process. To perform an early estimate, historical data are needed to fulfill the missing information with some statistical function. The fact is that the early estimates are far more valuable than the estimates in design time, but difficult to acquire in desired accuracy. In our approach, the statistical approach is used.

This paper is divided into five sections. In Section 2, the importance of size estimation is emphasized, the FPA method is briefly described, and a detailed review of the literature is given. In Section 3, different OO-to-FPA mappings are described and compared. Based on the results, a new, unified OO-to-FPA mapping is proposed in Section 4, together with several additional improvements that address current size estimation problems. The problem of inaccuracy in early estimates is addressed with the aid of different estimation levels. Section 5 summarizes the results of the improvements that were introduced into the size estimation process for object-oriented projects and also discusses further improvements.

2. Related work

The research community found several problems related to the FPA method. These problems can be grouped together by their research area:

1. Correlation between FPA elements [17,18,20].
2. Inappropriate formulation of the VAF and the General System Characteristic (GSC) [21].
3. Informal definition and violation of monotony [1,6,7].
4. The gap between OO and FPA abstractions [3–5,8, 27,31].

Lokan [20] analyzed 269 projects and tried to discover the correlation between five elements used in the FPA method to represent a software system. He found out that external inputs (EI) and internal logical files (ILF) always correlate, while external interface files (EIF) rarely correlate with other elements.

Another paper [21] analyzed 235 projects with an emphasis on GSC. The research was divided into two parts. In the first part, Lokan proved that GSC is out-of-date and not appropriate for today's systems. In the second part, he used empirical analysis to show that 14 technical factors were not independent and expressed overlapping characteristics. Therefore, the smaller set could be used instead. The research also showed that VAF improved the estimated size in less than half of all cases.

All methods for software size estimation lack adequate formal foundations in their origin descriptions. There were some attempts [6,7] to add formality to functional size measurement. Fetcke's model is applicable to different methods, since it introduced an additional level of abstraction, called a data-oriented abstraction. The approach proposed by Diab et al. was designated COSMIC-FPP [35] and had a specific purpose. In our research, the model defined by Fetcke is used as a basis and further refined by the definition of a mapping function [33]. The abstraction is also used to formally describe an OO-to-FPA transformation, as proposed in this paper.

The violation of monotony first addressed in MK II FPA [30], later formally proven by Fetcke [6] and again addressed in research [1] resulted in a change in complexity weights. Since our research also resulted in a change of complexity tables different from the one proposed by Al-Hajri et al. [1], this approach is discussed in more detail. In Al-Hajri's research, tables gathered with the training methods from neural networks replaced the original complexity tables. The results showed that the average error decreased and the convergence between actual effort and estimated effort improved. The authors extended complexity tables and related ordinal scales with absolute ones. However, the new tables are not appropriate for our research, since we deal with object-oriented systems that have specific characteristics. These characteristics will be presented in Section 3. We believe that Al-Hajri's approach could also be used to further improve our tables. Unfortunately, at the moment there is not enough empirical data available in the ISBSG repository [11] to apply his procedure of weights validation.

3. OO-to-FPA mappings

3.1. Method proposed by Fetcke et al.

Fetcke et al. [8] focused their research on a specific method, namely object-oriented software engineering