The Journal of Systems and Software 84 (2011) 976–984

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Function point measurement from Web application source code based on screen transitions and database accesses

T. Edagawa[a,1], T. Akaike[a,2], Y. Higo[a], S. Kusumoto[a,*], S. Hanabusa[b], T. Shibamoto[b]

[a] Graduate School of Information Science and Technology, Osaka University, Suita, Osaka, Japan
[b] Hitachi Systems & Services, Ltd., Otaku, Tokyo, Japan

ARTICLE INFO

Article history:
Received 7 April 2010
Received in revised form
12 December 2010
Accepted 13 January 2011
Available online 25 January 2011

Keywords:
Function point
Estimation
IFPUG
Web application
Empirical Software Engineering

ABSTRACT

A function point (FP) is a unit of measurement that expresses the degree of functionality that an information system provides to a user. Many software organizations use FPs to estimate the effort required for software development. However, it is essential that the definition of 1 FP be based on the software development experience of the organization. In the present study, we propose a method by which to automatically extract data and transaction functions from Web applications under several conditions using static analysis. The proposed method is based on the International Function Point Users Group (IFPUG) method and has been developed as an FP measurement tool. We applied the proposed method to several Web applications and examined the difference between FP counts obtained by the tool and those obtained by a certified FP specialist (CFPS). The results reveal that the numbers of data and transaction functions extracted by the tool is approximately the same as the numbers of data and transaction functions extracted by the specialist.

1. Introduction

In the planning of software projects, software size is an important factor for estimating the development effort or development period. The function point (FP) (Albrecht, 1994) is a software size metric that is widely used in business application software development. Since FPs measure the functional requirements, the measured software size remains constant regardless of the programming language, design technology, or development skills involved. In addition, when planning development projects, FP measurement can be applied early in the development process. A number of FP methods have been proposed. The International Function Point Users Group (IFPUG) method and the COSMIC method have been widely used in software organizations.

However, a number of problems with these methods remain to be solved. One such problem is benchmarking (IFPUG, 2002). If an organization introduces FP for development effort or development period estimation, base data must be collected. That is, FPs must be measured on the basis of software that was developed previously by the organization using a typical development process. Usually, FPs are counted from design specifications. However, during development, a number of functionalities are frequently added and modified. Therefore, the actual functionalities exist only in the source code and counting FPs from source code is costly. Moreover, differences may occur for the same product, even in the same organization, because FP measurement involves judgment on the part of the counter (Kitchenham, 1997). For example, Low and Jeffery (1990) reported that a 30% variance was generated within one organization and a variance of over 30% is generated across organizations. One promising approach to solving these problems is to automate the FP measurement.

Several studies have investigated automation of FP measurement from design specifications. For example, Diab et al. (2002) proposed a formalization of the IFPUG FP definition for automated measurement of B specifications. Lamma et al. (2004) presented a tool for FP measurement from the specifications of a software system expressed in the form of an Entity Relationship (ER) diagram and a Data Flow Diagram (DFD). Cantone et al. (2004) considered the convertibility of the elements of the UML into entities of the FPA, introduced a model for establishing the link between these, and conducted a pilot study comparing the FP count obtained by the model with that obtained by a CFPS. We previously proposed a method by which to automate FP measurement from requirements specifications and UML design specifications (Kusumoto et al., 2000; Uemura et al., 2001). Automating FP measurement

* Corresponding author at: Department of Computer Science, Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan. Tel.: +81 6 6879 4110; fax: +81 6 6879 4114.
E-mail addresses: higo@ist.osaka-u.ac.jp (Y. Higo), kusumoto@ist.osaka-u.ac.jp (S. Kusumoto), shigeo.hanabusa.zg@hitachi-solutions.com (S. Hanabusa), toshihisa.shibamoto.hz@hitachi-solutions.com (T. Shibamoto).
[1] Currently, he belongs to Tecmo Koei Holdings Co., Ltd., Japan.
[2] Currently, he belongs to Hitachi, Ltd., Japan.

0164-1212/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2011.01.029

from these specifications is pragmatic in the early phase of software development.

We have investigated automatic FP measurement for object-oriented software in collaboration with Osaka University and Hitachi Systems & Services (Kusumoto et al., 2000, 2002; Uemura et al., 2001). In Kusumoto et al. (2002), we attempted to measure the FP count on the basis of the execution trace data of a Java program. However, it is necessary to prepare a sufficient number of test cases that execute all of the functionalities of the program. In addition, the accuracy of the FPs measured by this method was not high. Thus, the method was not adequate for practical application in the context of Hitachi Systems & Services.

In the present paper, we intend to examine the possibility of automatic FP measurement from source code by static analysis. First, we propose a measurement method by which to count data and transaction functions from Web application source code under several conditions. We then develop an FP measurement tool based on the proposed method. Finally, we apply the developed tool to seven independently developed Java programs with the same specifications and examined the difference between the FP values obtained by the tool and the FP values obtained by a CFPS at Hitachi Systems & Services. The results represent the applicability of the proposed method.

The contributions of the present paper are as follows:

- The present paper describes a concrete FP measurement procedure for a typical Web application software.
- An FP measurement tool based on the proposed method is developed.
- The results of case studies conducted for several Web applications are presented. These applications are not commercial software applications but have the characteristics of actual software. Finally, the validity of these results is confirmed.

The remainder of the present paper is organized as follows. Section 2 presents an overview of FP analysis and the IFPUG method. The proposed method is explained in Section 3. A brief explanation of the FP measurement tool based on the proposed method is given in Section 4. Section 5 presents the results of the case studies and a discussion of the results. Section 6 describes related research, and Section 7 discusses the strengths and weaknesses of the proposed method. Finally, the conclusions and areas for future study are presented in Section 8.

2. Function point analysis

2.1. Overview

An FP is a unit of measurement that expresses the degree of functionality provided by software. FPs can be determined from the requirements specifications, the design specifications, and the program code. Unlike Lines of Code (LOC), since the FP measures functionality, the FP is said to be independent of the technology and programming language used for the software implementation.

Albrecht (1994) proposed the original FP analysis. Since then, several types of FP analysis methods, such as the IFPUG method, the COSMIC method, 3D FPs (Jones, 1996), Feature Points (Jones, 1996), and MarkII (Symons, 1991), have been proposed. In the present paper, we focus on the IFPUG method, because, compared to other methods, this method provides detailed procedures and rules for FP counting and has been widely applied to business application software.

2.2. IFPUG method

The IFPUG method is a modified version of Albrecht's FP method. In the IFPUG method, the complexity of the software is objectively established and the rules of the counting procedures are described in detail.

Table 1
RET/DET complexity matrix.

RET	DET		
	1–19	20–50	51–
1	Low	Low	Average
2–5	Low	Average	High
6–	Average	High	High

In the IFPUG method, the FP counting procedure consists of seven steps. Step 1: Determine the type of FP counting, Step 2: Identify the counting boundary, Step 3: Count the data function types, Step 4: Count the transaction function types, Step 5: Determine the unadjusted FP count, Step 6: Determine the value adjustment factor, and Step 7: Calculate the final adjusted FP count.

In the following, we explain Steps 2, 3, 4, and 5 in order to clarify the rules proposed in Section 3.

Step 2 (Identify the counting boundary): A boundary indicates the border between the application or project being measured and the external applications or the user domain. A boundary establishes which functions are included in the FP count.

Step 3 (Count the data function types): Data function types represent the functionality provided to a user to meet internal and external data requirements. Data function types are classified into the following two types: internal logical files (ILFs) and external interface files (EIFs).

Data functions are defined as follows:

Internal logical file (ILF): (1) The group of data is a user-identifiable group of data. (2) The group of data is maintained within the application boundary. (3) The group of data identified has not been counted as an EIF for the application.

External interface file (EIF): (1) The group of data is a user-identifiable group of data. (2) The group of data is not maintained by the application being counted. (3) The group of data identified has not been counted as an ILF for the application.

Here, the term "file" refers to a logically related group of data and not to the physical implementation of this group of data.

Then, assign a functional complexity to each identified ILF or ELF based on the number of data element types (DETs) and record element types (RETs) associated with the ILF or EIF using the RET/DET complexity matrix (see Table 1). A DET is a unique user-recognizable, non-recursive field on the ILF or EIF. An RET is a user-recognizable subgroup of data elements within an ILF or EIF.

Step 4 (Count the transaction function types): Transaction function types represent the functionality provided to a user for the processing of data by an application. There are three transaction function types: external input (EI), external output (EO), and external inquiry (EQ). These transaction functions are defined as follows:

External input (EI): An external input processes data or control information that comes from outside the application's boundary. The external input itself is an elementary process.

External output (EO): An external output is an elementary process that generates data or control information sent outside the application's boundary.

External inquiry (EQ): An external inquiry is an elementary process made up of an input–output combination that results in data retrieval. The output contains no derived data. Here, derived data is data that requires processing other than direct retrieval and editing of information from internal logical files and/or external interface files. No internal logical file is maintained during processing.