On an Automatic Function Point Measurement from Source Codes

Shinji Kusumoto, Takuto Edagawa, Yoshiki Higo
*Graduate School of Information Science and Technology, Osaka University
{kusumoto, t-edagw,higo}@ist.osaka-u.ac.jp

Abstract

This paper describes issues of introducing function point analysis to software organizations and then proposes a method to automatically extract data and transaction functions from Web application using static analysis.

1. Introduction

In global software development projects, it is important to quantitatively grasp the progress of each software development block. Size of software is sometimes used to check the progress of the development. Function point (FP) [1] is one of the software size metrics and it has been widely used in business application software developments. Because it measures the functional requirements, the measured size stays constant despite the programming language, design technology, or development skills involved. Also, it is available in the early stage of the development process, for planning design and development projects. Up to the present, various FP methods have been proposed. The IFPUG (International Function Point Users Group) method has been widely used in software organizations.

However, several problems still remain. One of them is benchmarking. If an organization tries to introduce FP for estimation, it is necessary to collect the base data. That is, FPs have to be measured from the past software developed there. Usually, FP is counted from design specifications. But, during the development, some functionalities are frequently added and modified. So, the actual functionalities exist only in the source code and counting FP from source code is cost-consuming. Moreover, it is reported that differences for the same product may occur even in the same organization since FP measurement involves judgment on the part of the counter. One of the promising approach to solve the problems is to automate the FP measurement.

This paper describes the conventional approaches to automate the FP measurement and the issues. Then, we briefly explain our approach under industry-university cooperation. In this paper, we intend to examine the possibility to measure FP from source code automatically using static analysis: At first, under several conditions, we propose a measurement method to count data and transaction functions from Web application source code.

2. Function Point Analysis

Function point (FP) measures the functionality provided by software. It can be determined from the requirements specification, design specification and program code. Unlike LOC, since FP measures functionality, it is said to be independent of the technology and programming language used for the software implementation. Allan Albrecht first proposed original function point analysis. The IFPUG method is a modified-one of the Albrecht's function point. In the modification, the evaluation of the complexity of the software was objectively established and the rules of the counting procedures were also described minutely and precisely. In the IFPUG method, the counting procedure of FP consists of seven steps[2], Step1: Determine the Type of Function Point Count, Step2: Identify the Counting Boundary, Step3: Count Data Function Types, Step4: Count transaction Function Types, Step5: Determine the Unadjusted Function Point Count, Step6: Determine the Value Adjustment Factor and Step7: Calculate the Final Adjusted Function Point Count.

3. Conventional Approaches

An appropriate method to measure FP from source codes has been proposed in [3]. The method firstly transforms source codes to specifications written by formal notations and then calculates FP based on the IFPUG method from the specifications. However, to our knowledge, actual application results have not been reported. It would be difficult for software organizations to develop such translators in the actual software development under several limitations.

There is another way to estimate the development effort by using some metrics. In [4], a new complexity metric, called DataBase Points (called DBP) has been proposed. DBP is a point calculated based on items related to database (table, relation, transaction, form, reports). DBP estimates accurate effort, even it can apply to only the system using MS-ACCESS. Since the key idea of our method is also extraction of tables and transactions, measurement of DBP is similar to ours. But, the definition of DBP is not based on FP.

4. Our approach

Considering the automatic FP measurement from source code, it is not realistic to propose a method that can be applicable to any software. In this paper, we set the target application to kinds of Web applications using SQL database systems. The main steps of FP measurement are identifications of data and transaction functions included in the target software. Data function (DF) is a set of logically meaningful data, and it is used by transaction functions (TFs). Thus, we consider DFs as database tables used in the application. On the other hand, TF is an input-output processing to DFs and so we regard TF as a set of SQL statements executed in a screen transition.

In order to extract DFs and TFs, we introduce SQL tree. SQL Trees are constructed by analyzing the source code implementing Actions. An SQL Tree is a data structure storing SQL statements, and it includes a branch structure of the source code. The tree is composed of three types of nodes: 'Code Node', 'Branch Node' and 'SQL Call Node'. Code Nodes correspond with source code range, and Branch Nodes correspond with conditional branch, and SQL Call Nodes correspond with SQL calls. Then, using SQL trees, DFs and TFs are extracted. We regard a DF as a database table used by target application. These tables can be extracted by analyzing SQL Trees. After that, each of DFs is classified into ILF or EIF as follows:

(D1): If its table is accessed by 'insert', 'update', or 'delete', the database state is changed. Therefore, it is regarded as ILF,

(D2): If RuleD1 is not satisfied, it is regarded as EIF.

A TF is identified as a set of SQLs which can be executed simultaneously in the Action. Then, using the following rules, each of TFs is classified into EI, EO and EQ.

(1): If the TF has at least one SQL statement which changes database state, it is regarded as EI,

(2): If RuleT1 is not satisfied, at least one data obtained from database is converted, it is regarded as EO,

(3): If both RulesT1 and T2 are not satisfied, it is regarded as EQ.

5. Measurement Tool

We have developed a FP measurement tool. The target application is supposed to satisfy the following conditions:

- a main logic is written by Java,
- uses Struts framework,
- accesses to the database by JDBC or iBATIS .

It is the reason why the application is supposed to be developed by Struts that the extraction of screen transition is easy. Because, in Struts, screen transitions are determinably defined by configuration files. Since our proposed method doesn't include the extracting method of screen transition, it depends on the tool. The developed tool cannot validate whether or not the proposed method can apply the application developed without Struts. However, no matter how the application is developed by some methodology, the application has to define correspondence relationship of a user action and screen transition. Therefore, we believe that screen transition is able to be extracted easily. As for a database access method, the present tool is intended for JDBC or iBATIS. But, since the information of the called SQL is only needed, we can easily deal with an application using other access methods. The tool mainly consists of the parser and the FP calculator. The parser firstly analyzes source code(Java file and configuration files) and provides the information needed to calculate FP. Then, FP calculator extracts TFs and DFs with the information obtained by the parser, and calculates FP.

6. Conclusions

In this paper, we have proposed FP measurement by focusing screen transitions in Web application and database accesses. TFs are extracted from screen transitions as a set of SQLs. DFs are extracted from SQLs as a database table. In the future, it is necessary to improve the proposed method and the tool through continuous case studies/experiments.

Acknowledgment

This work is being conducted as a part of Stage Project, the Development of Next Generation IT Infrastructure, supported by Ministry of Education, Culture, Sports, Science and Technology and Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (C) (20500033).

References

[1] A. J. Albrecht. Function point analysis. *Encyclopedia of Software Engineering*, 1:518–524, 1994.

[2] International Function Point Users Groupg. *Function Point Counting Practices Manual Release 4.2*, 2004.

[3] A. April, E. Merlo, and A. Abran. A reverse engineering approach to evaluate function point rules. In *Fourth Working Conference on Reverse Engineering*, 1997.

[4] S. Abiad, R. Haraty, and N. Mansour. Software metrics for small database applications. In *ACM symposium on Applied computing*, volume 2, pages 866–870, 2000.