Journal of Engineering Technology

Volume 7, Special Issue (Internet of Things),Oct. 2018, PP. 246-261

RESEARCH ARTICLE -ENGINEERNG TECHNOLOGY ASEE

Validation of the Cost Estimations with Object Oriented Function Point (FP) through Software Visualization

So Young Moon, Byungkook Jeon* and R. Young Chul Kim

SE Lab., Dept. of Software and Communications Engineering, Hongik University, Sejong Campus, 30016, Korea

*Dept. of Software, Gangneung-Wonju National University, Wonju City, Gangwon Prov. 26403, Korea

Abstract: In the starting point of software development projects, it is difficult to estimate software size due to the invisibility and no development of software. In addition, it may be incurred different costs per changing the size of a project depended on the frequent changes of requirements during the project. The previous approaches have focused on the cost estimation that computes the size of a project with the function point (FP), the object oriented function point (OOFP), or the use case point (UCP) method, but never validated it. To solve this problem, we suggest our visualized method to validate whether it may be the approximate cost or not. This approach is a code visualization based on reverse engineering, which can validate the cost estimation of any project through the final source code developed. Through this method, we may judge the cost legitimation before/after a project.

Keywords: Function Point, Reverse Engineering, Validating Cost Estimation, Code Visualization

1 Introduction

As the huge use of software, it is emerging the important issue of the cost estimation of software development and effective cost management. Software engineering include technologies, methodologies, and tools that facilitate in developing high quality software on frequently changing requirements within a given period. From a manager's perspective, to estimate the period, cost, and efforts depended on the software size is important to work a successful project for high-quality software development. It is difficult, however, to estimate the cost for software development due to vague requirements and a lack of the software development's history data [1].

In some cases, the client may estimate the software development cost and the development company plans the project according to the cost. The function point has been most widely used for estimating the software size, which has applied to estimate the project development cost and mitigated the shortcomings [2], [3]. Some studies also improve the function point to predict the development costs [4], [5]. However, their results were far from the reality because they did not measure and validate to spend a reasonable cost with the developed software, i.e., the source code.

To solve this problem, we suggest validating the cost estimation based on the source code that is latest output by software development. We also propose the visualizing tool with the source code based on a reverse engineering method, which identify the function types required to calculate the function point, and measure the function complexity.

The section 2 describes the related works. The section 3 mentions the validation method based on reverse engineering. The section 4 shows how to apply with an automobile goods management software program as an example. Finally, it makes conclusions and future work.

2. Related Work

2.1 Function Point Mechanism

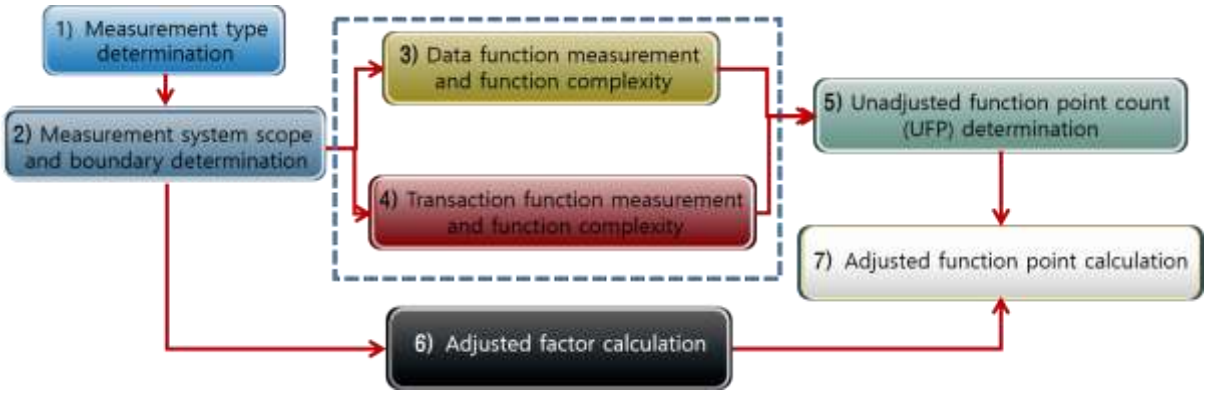

Figure. 1. Function point process of IFPUG

Figure. 1 shows the function point measurement process recommended by IFPUG as follows: 1) at the first step of measurement type determination, we determine the type to measure among a new development project, an improvement project, and an application. 2) At the second step of measurement system, we identify the scope and boundary of the elements to be included in the measurement scope, and determine the boundary between the measured application and the external users. 3) In the data function measurement step, we measure to distinguish about ILF (Internal Logical File) and EIF (External Interface File). An ILF is a data file that is stored, maintained, and modified within the application boundary, and an EIF is a data file referenced from the outside of the application boundary. As for the measurement method, identification is performed according to the aforementioned ILF and EIF measurement rules, and "low," "average," or "high" is determined using ① of Table 1 according to the measurement rules about DET(Data Element Type) and RET(Record Element Type). 4) In the transaction function measurement step, external input (EI), external output (EO), and external inquiry (EQ) are identified according to measurement rules. Using K in Table 1, the functional complexity for EI, EO, and EQ is calculated as low, average, and high according to the value of DET / FTR. 5) In the unadjusted function point (UFP) count determination step, we add the data function and transaction function of each function that are calculated in the previous data function measurement and transaction function measurement steps. The UFP is measured by measuring the data and transaction functions and using ③ of Table 1 for the ILF, EIF, EI, EO, and EQ.

Table 1. Complex and Weight point of DET, RET, and FTR

		Data Element Type		
		1–19	20–50	>=51
Record Element Type	1	Low	Low	Average
	2–5	Low	Average	High
	>5	Average	High	High

		Data Element Type		
		1–19	20–50	>=51
File Type Referenced	1	Low	Low	Average
	2–5	Low	Average	High
	>5	Average	High	High

		Function Levels		
	Components	Low	Average	High
	ILF	x7	x10	x15
	EIF	x5	x7	x10
	EI	x3	x4	x6
	EO	x4	x5	x7
	EQ	x3	x4	x6

2.2 Code Visualization through Reverse Engineering

Forward engineering sequentially performs the requirement analysis, design, and implementation steps. It includes a requirements specification, designs and implements through an upper level abstract concept analysis. Meanwhile, reverse engineering produces the outputs of the previous step reversely based on the code outputs, and goes back from the implementation to design and from the design to requirements [6]. In general, the source code is used as the input of the reverse engineering process, and the purpose of reverse engineering is to derive the design and specification of the system from the source code. Reverse engineering is utilized for a better understanding of the system in many cases and is used as part of the re-engineering process. Reverse engineering restores the design that facilitates in understanding the created software [7].

In this study, a code visualization mechanism is used to validate the function point through reverse engineering. Software visualization is a technology to manage the software quality, and improve its maintenance by visualizing the source code and development process proposed by NIPA in Korea. For high quality software development, we need to choose the development, test automation, and quality certifications such as CMMI, TMMi, and SP. However, IT venture companies, small and midsize companies do not even attempt to acquire such quality certifications owing to their high costs. The software visualization of NIPA facilitates such companies generally constrained by the lack of human and financial resources to develop high quality software [8]. Table 2 shows to compare the benefits of forward engineering and reverse engineering obtained through code visualization. From a forward engineering perspective, it is possible to review the code during development, identify a code

progress rate for each developer, and design the test driver/stub in advance. From a reverse engineering perspective, refactoring is easy in the absence of the previous developer, and it is possible to realize low complexity through code complexity visualization, such as coupling and cohesion. Furthermore, understanding the legacy software, we can enhance code with the reused module and design extracted.

Table 2. Comparison between reverse engineering and forward engineering

Forward Engineering	Reverse Engineering
① Code review is possible during development ② Code progress rate for each developer can be identified ③ Test driver/stub can be designed in advance	① Easy refactoring in the absence of the previous developer ② Code complexity can be visualized (coupling and cohesion) ③ Reuse module unit extraction ④ Design extraction ⑤ Architecture extraction ⑥ Performance extraction

Figure. 2 shows the tool chain for code visualization. While the existing tool chain [8], [10], [12] used the source navigator, it was replaced with a Java development tool (JDT)-based Java parser in this study. The reason is described in section 3.3. Listed below is the execution process of the tool chain.

1) JDT-based Java parser parses the source code.
2) Parsed code syntax is stored in the SQLite database.
3) Data are retrieved from the database by creating a query statement that corresponds to the ILF, EIF, EI, EO, and EQ related to the function point, and a dot script is created using the retrieved contents. In this step, visualization is possible by creating a query statement for coupling and cohesion, reuse module unit extraction, design extraction, architecture extraction, and performance extraction.
4) Visualization is performed using the contents of the dot script, and the visualized contents are different depending on the query statement created in 3). Therefore, it is possible to extract the coupling graph, class diagram, sequence diagram, and use case diagram.

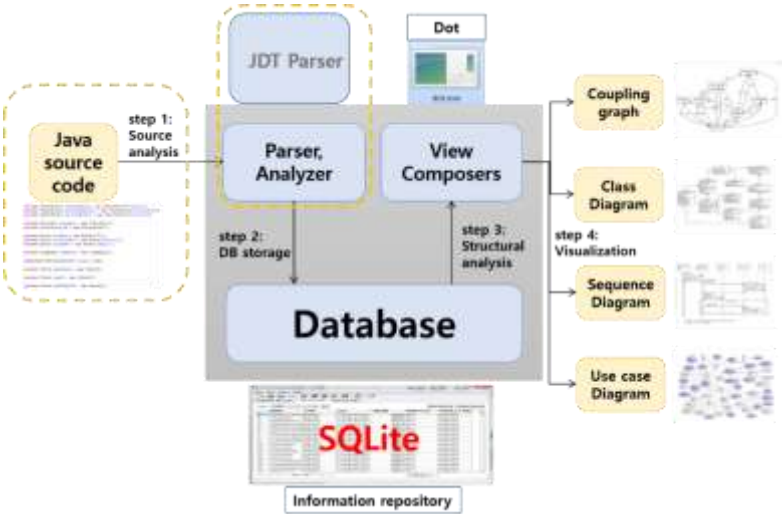

Figure. 2. Tool chain with JDT-based parser for code visualization

3. Validation for Cost Estimation based on Reverse Engineering

In Korea domestic software cost estimation guidelines, most companies use a simplified method as follows: 1) determines the measurement type of a function point, 2) identifies the calculation scope and application boundary, 3) calculates the data function and complexity, 4) calculates the transaction function and complexity, and then 5) calculates the UFP. We attempt to divide the data function and transaction function types in the dotted line among the function point measurement processes recommended by IFPUG. Figure. 1 shows to calculate the complexity of each function through visualizing source code.

In order to calculate function points, you need to know about data functions and transaction functions. In this chapter, we investigate data and transaction functions according to the regular method recommended by IFPUG, and calculate the complexity by extracting the number of functions.

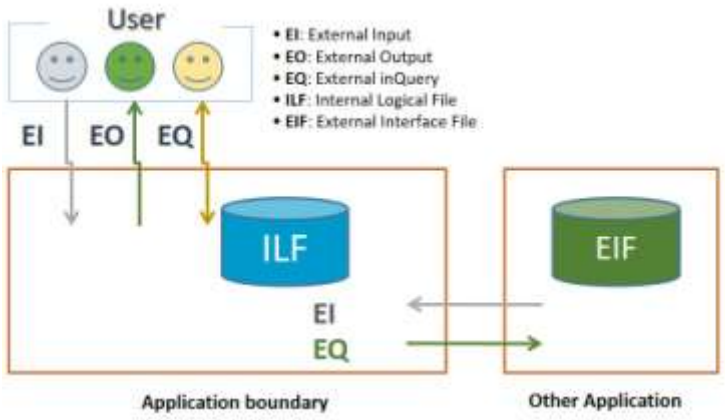

Figure. 3. Data and transaction types

Figure. 3 shows the data and transaction function types. The data function types must find the ILF and EIF, and extract the number of functions. ILF refers to the logical data that the user knows or a logical table that is actually maintained inside the application as the control information. The EIF is a logical data group or control information that the user knows, and refers to a logical table that is not updated or added within the application under development but is referenced by external applications. The transaction function types refer to the registration, modification, deletion, inquiry, and output functions, which are provided to the user by the application to satisfy the data processing requirements. Three function types have EO, EQ, and EI. 1) EO sends data to the outside of the current system's boundary. Unlike data inquiry, it provides information to the user through another processing logic (mathematical formula application and calculation processing) after an inquiry. 2) EQ refers to a function that reads an ILF and shows the data to the outside of the application boundary. 3) EI means a function that modifies, inputs, and deletes data from the outside to the inside of the application by the user or system. The process for obtaining the data and transaction functions from the source code is as follows.

3.1 The Classification of Functional Requirements from Client Requirements

The definition and classification of functional requirements from client requirements must be as shown in Table 3. There are 17 functions defined by the EI, EO, EQ, or ILF function types. Table 3 shows the function definition of functional requirements.

Table 3. The Function Definition of Functional Requirements

No	Task	Function Name	Function Type
Function 1	Login	Login	EQ
Function 2	Customer Management	Customer Register	EI
Function 3		Customer Update	EI
Function 4		Customer Retrieve	EQ
Function 5		Customer Delete	EI
		Customer Table	ILF
Function 6	Sale Management	Sale Register	EI
Function 7		Sale Update	EI
Function 8		Sale Retrieve	EQ
Function 9		Sale Delete	EI
Function 10		Sale Save as Excel	EO
		Sale Table	ILF
Function 11	Stock Goods Management	Stock Register	EI
Function 12		Stock Update	EI
Function 13		Stock Retrieve	EQ
Function 14		Stock Delete	EI
Function 15		Stock Save as Excel	EO
		Stock Table	ILF
Function 16	Income/Outcome Management	Income/Outcome Retrieve	EO
Function 17		Income/Outcome Save as Excel	EO

As the method-naming coding rule is applied, the "transaction function method" is used for the method name of the source code for which each function is applied. In addition, the query statement is created with the variable name "query" inside of the method. For example, in the case of the "customer register" of function 2, the method name is written as "EI_registerCustomer". Furthermore, in the case of the query statement, the variable name "query" must be writtern in the defined method to calculate the DET. For example, [String query=" insert into customer values(name, phone, carNumber, carType);"] must be written.

3.2 Function Point Calculation through Code Paring

- **Parsing Source code through JDT based parser**

We perform to visualize the inner code structure with the tool chain for code visualization proposed by an existing study [8]. The source navigator conducts code syntax parsing, but is limited to the class, function, variable, method, and structure of its analyzing scope, thereby rendering it impossible to perform visualization by analyzing all the sentences of Java. As it does not perform parsing for all the statements inside the method necessary for our study, its function is replaced by a developed JDT-based Java parser [9].

- **Database for Function Point**

The ILF and EIF used in each function are obtained through the tool chain, and the number of functions is extracted by obtaining the transaction function used in each function. Figure. 4 shows the method information in the database.

DBManager	method	public	1	0	boolean	EI_customerReg	String, String, String, String
DBManager	method	public	1	0	boolean	EI_customerDelete	String, String, String, String
DBManager	method	public	1	0	ArrayList<Ha···	EQ_customerSearch	String, String
DBManager	method	public	1	0	ArrayList<Ha···	EQ_customerSellSearch	String, String
DBManager	method	public	1	0	HashMap<St···	EQ_login	String, String
DBManager	method	public	1	0	void	EI_customerMod	String, String, String, String
DBManager	method	public	1	0	void	EI_sellReg	String, Date, String, String···
DBManager	method	public	1	0	void	EI_sellMod	String, Date, String, String···
DBManager	method	public	1	0	void	EI_sellDel	String, String, String
DBManager	method	public	1	0	ArrayList<Ha···	EQ_sellSearch	Date, Date
DBManager	method	public	1	0	void	EI_wearReg	String, String, int, String, int···
DBManager	method	public	1	0	void	EI_wearMod	int, String, String, int, St···
DBManager	method	public	1	0	void	EI_wearDel	int
DBManager	method	public	1	0	ArrayList<Ha···	EQ_wearSearch	Date, Date
DBManager	method	public	1	0	HashMap<St···	EQ_sellWearSearch	Date, Date

Figure. 4. Method information in database

- **Function Point Calculation**

The total function points are calculating with the function complexity with the extracted data and transaction functions. As determining the transaction function among the EI, EQ, and EO, the functions obtained through the applied coding rule corresponding to the ILF and EIF are found on the query statement used by each function. Table 4 shows the list of the extracted queries used by each function. Currently, the queries are processed through a string split, and are entered into the database, which will be processed by an SQL parser to develop in the future. We can calculate the complexity by obtaining the number of fields used in each query statement in the database. With this method, the *customer registration function* is 1) that the function type is EI, one reference file type is a customer table, and 2) that four data element types, one function key, and two dialogs. As a result, we can say that function complexity is low, and the function point is three.

Table 4. Used query statement for function

Task/Table	Function Name	Query

Customer Management /Customer	Customer Register	String query = "insert into **customer** values(?,?,?,?)";
	Customer Update	String query = "update **customer** set carnum=?, carcat=? where customer_name=? and phonenum=?";
	Customer Retrieve	String query = "select CUSTOMER_NAME, PHONENUM, carnum, carcat from **customer** where customer_name LIKE ? AND phonenum LIKE ?";
	Customer Delete	select CUSTOMER_NAME, PHONENUM, carnum, carcat from **customer** where customer_name LIKE ? AND phonenum LIKE ?
Sale Management /sale	Sale Register	String query = "insert into **sale** values(?,?,?,?,?,?,?,?,?,?)";
	Sale Update	String query = "update **sale** set SALE_DATE=?, SALE_DETAIL=?, SALE_MODEL=?, CAR_PART=?, PRICE=?, PAYMENT=?, SALESPERSON=? " + "where CUSTOMER_NAME=? AND CUSTOMER_PHONENUM=? AND SALE_NAME=?";
	Sale Retrieve	String query = "select CUSTOMER_NAME, CUSTOMER_PHONENUM, SALE_DATE, SALE_NAME, SALE_DETAIL, SALE_MODEL, CAR_PART, PRICE, PAYMENT, SALESPERSON " + "from **sale** " + "where SALE_DATE between TO_DATE(?,'MM dd,yyyy HH24:MI') " + "and TO_DATE(?,'MM dd,yyyy HH24:MI')";
	Sale Delete	String query = "delete from **sale** where CUSTOMER_NAME=? AND CUSTOMER_PHONENUM=? AND SALE_NAME=?";
Stock Management /Stock_goods2	Stock Register	String query = "insert into **stock_goods2** values(GOODS_ID_SEQ.NEXTVAL,?,?,?,?,?,?,?,?)";
	Stock Update	String query = "update **stock_goods2** set GOODS_SPEC=?, GOODS_AMOUNT=?, GOODS_UNIT=?, UNIT_PRICE=?, TOTAL_PRICE=?, STOCK_DATE=?, NOTE=?, SALE_PRICE=?, GOODS_NAME=? " + "where GOODS_ID = ?";
	Stock Retrieve	String query = "select GOODS_ID, GOODS_NAME, GOODS_SPEC, GOODS_AMOUNT, GOODS_UNIT, UNIT_PRICE, TOTAL_PRICE, STOCK_DATE, NOTE, SALE_PRICE " + "from **stock_goods2** " + "where STOCK_DATE between TO_DATE(?,'MM dd,yyyy HH24:MI') and TO_DATE(?,'MM dd,yyyy HH24:MI')";
	Stock Delete	String query = "delete from **stock_goods2** where GOODS_ID = ?";
Income and	Income/Outcome	String querySell = "select SALE_DATE, PRICE "

Outcome /**Sale stock_goods2**	Retrieve	+ "from **sale** " + "where SALE_DATE between TO_DATE(?,'MM dd,yyyy HH24:MI') and TO_DATE(?,'MM dd,yyyy HH24:MI')"; String queryWear = "select STOCK_DATE, TOTAL_PRICE " + "from **stock_goods2** " + "where STOCK_DATE between TO_DATE(?,'MM dd,yyyy HH24:MI') and TO_DATE(?,'MM dd,yyyy HH24:MI')";

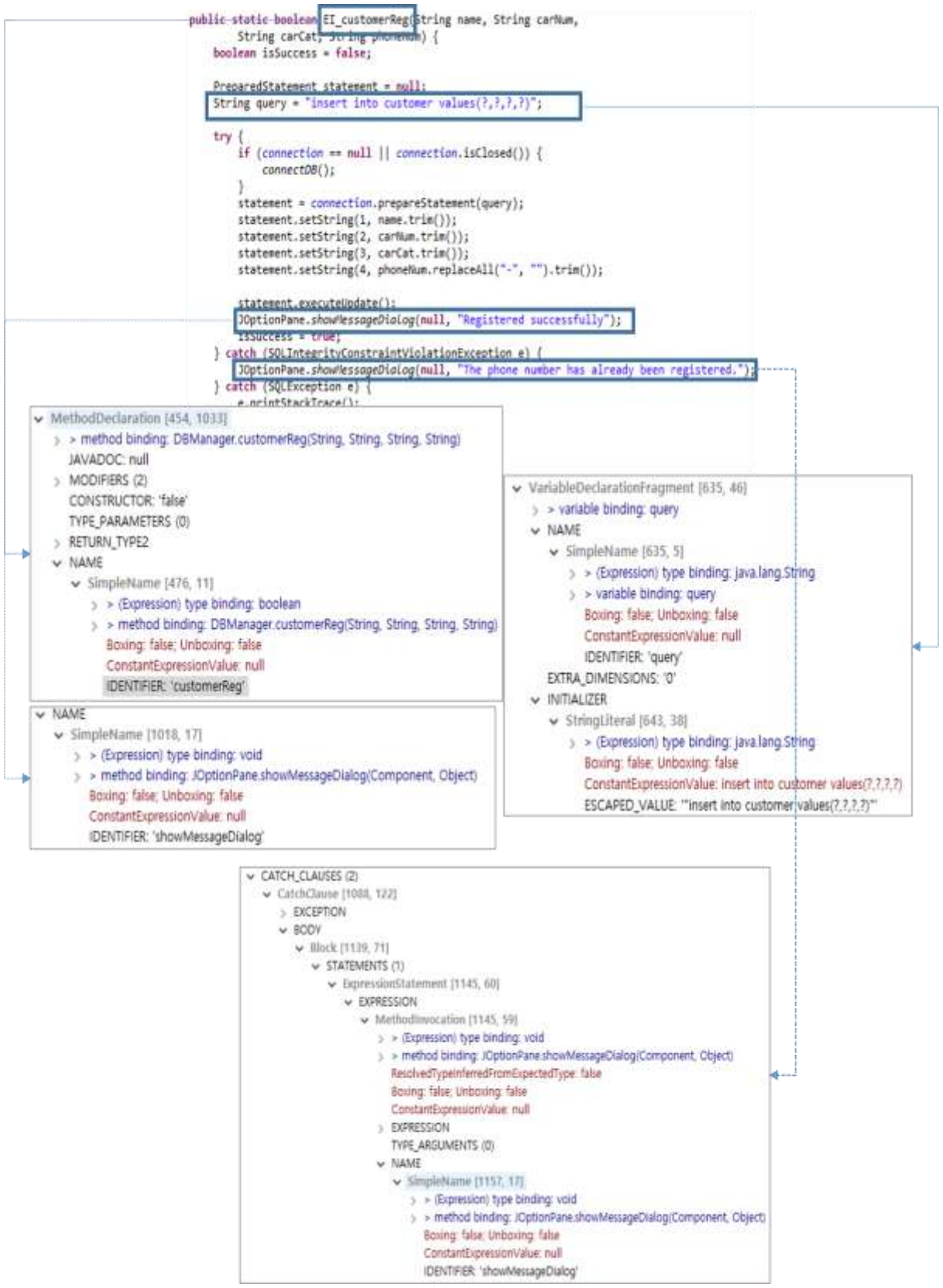

Figure. 5. Visualization method of "customer registration function"

4. Visualization of Target Program

The target program is a GUI environment developed with the Java language and an application that operates in connection with the Oracle database. It is a small-sized application with 17 functions. Actually we estimate cost of the target program by KRW 31,183,330 in Table 5. We try to validate this is reasonable cost by visualization technic based on reverse engineering. We analyze the target program through the tool chain and visualize as a class diagram, as shown in Figure. 6. We calculate function points through the source code associated with the class diagram in Figure 6. Table 5 is the estimated cost calculated manually according to the functional score calculation based on the customer's requirements. Table 6 shows the calculation of function points through the reverse engineering based code visualization proposed in this paper. The number of functions was 17, the customer registration of the customer management task corresponded to the EI. If you look closely at the expanded DB class in Figure. 6, the EI_customerReg method implements the function of customer registration. Percentage data by a function type presents in a left side of pie-chart in Figure. 6. The ratio of the target program is as follows. EQ is 16%, EI is 36%, EO is 20%, ILF is 28%. The ratio of EI is the highest at 36%, which means that there is a lot of data input among the functions through code visualization. Percentage data by a set of function presents in a right side of pie-chart in Figure 6. A set of login function is 4%, a set of income/expenses management function is 11%, a set of customer management function is 25%, a set of stock management is 29%, and a set of sale management is 31%. The right pie chart in Figure 6 shows that a set of sale management function has the highest work amount through code visualization. As middle side of pie-chart of Figure 6, we find out the highest function complexity is the sale management function.

In a previous study [11], we classified the function type by a code comment in a method declaration, but we improve a classifying program by applying the method-naming rule owing to problems in parsing and function extraction. As shown in Table 7, we classify into EI, EO, and EQ. We extract the method of each function, and measure the complexity and point for each function. As for Function15(EQ_SellSearch), the transaction type is EQ and one FTR is available. Furthermore, there are 12 DETs, including customername, phonenumber, installname, installdate, installcontent, installmodel, component, price, payment, master, funcBtn1, and message1, and the complexity was low, thereby resulting in the function point of three. The analysis of the source code inside the method for the function revealed that the customer table is referenced as an ILF and seven data element types are available, including the name, vehicle number, vehicle type, phone number, one function key (confirm/cancel key), and two dialogs. As a result of applying the IFPUG function point calculation method using the extracted values and the visualization tool chain developed in this study, the function complexity finds to be low and the function point is three. The application of the same measurement method to other functions revealed that the total function point for the data and transaction functions of the application is 75. Table 6 shows the development cost estimation after adjusting the target program. In this study, we use the software project cost estimation reference revised in 2017. We calculate the unit price per the analysis, design, implementation, and test steps of the software development life cycle. Using the calculation method for the analysis as an example, we calculate the analysis unit price as follows: (98,648) × total function point (75) × language (1.2) × application type (1.0) × size (0.65) × quality and feature (1.0) = KRW 5,770,908. The lower part of Table 1 shows the contents of the adjustment factors. The size adjustment factor is 0.65 because the function point is less than 300. The type adjustment factor is 1.0 for a business software. The language adjustment factor is 1.2 because the development is based on Java, and the quality and

feature adjustment factor is 1 because the total effect degree is determined to be zero from the distributed processing (0), performance (0), reliability (0), and multiple sites (0). The function point of the target program is 75 points through code visualization, and the total development cost was estimated to be KRW 30,373,374 in Table 6. We find a little difference between the estimated cost and the cost of implemented program by visualization. We make criteria for judgement about cost as (2). Also we define a formula as (1). We get a result value from (1), and compare by using (2). C is -2.6 percentage as (3). So we validate estimated cost is reasonable cost.

- *C: Radio of between calculated cost and estimated cost*
- *CCV: Calculated Cost by Visualization*
- *EC: Estimated Cost*

$$C = \frac{CCV\text{-}EC}{EC} \times 100 \qquad (1)$$

$$-10\% < C < 10\% \qquad (2)$$

$$C = -2.6\% \qquad (3)$$

Table 5. Estimated Cost of target program

Step	Cost per Step	Total Function Point	Adjusted Point				Cost
			Language	Application Type	Size	Quality & Feature	
Analysis	98,648	77	1.2000	1.00	0.6500	1.0000	5,924,798
Design	124,609						7,484,016
Implementation	166,145						9,978,668
Test	129,801						7,795,848
Total	519,203						**31,183,330**

Table 6. Calculated Cost by visualization based on reverse engineering for target program

Step	Cost per Step	Total Function Point	Adjusted Point				Cost
			Language	Application Type	Size	Quality & Feature	
Analysis	98,648	75	1.2000	1.00	0.6500	1.0000	5,770,908
Design	124,609						7,289,626
Implementation	166,145						9,719,482
Test	129,801						7,593,358
Total	519,203						**30,373,374**

	Type	Base	Adjusted Point
Adjusted Point	Size	0.108 × log e(Function Point) + 0.2229	**0.65**
	Type	Business Software	**1.00**
	Language	JAVA	**1.20**
	Quality & Feature	0.025 × Total Effect Degree(0) + 1	**1.00**

Journal of Engineering Technology

Volume 7, Special Issue (Internet of Things),Oct. 2018, PP. 246-261

RESEARCH ARTICLE -ENGINEERNG TECHNOLOGY ASEE

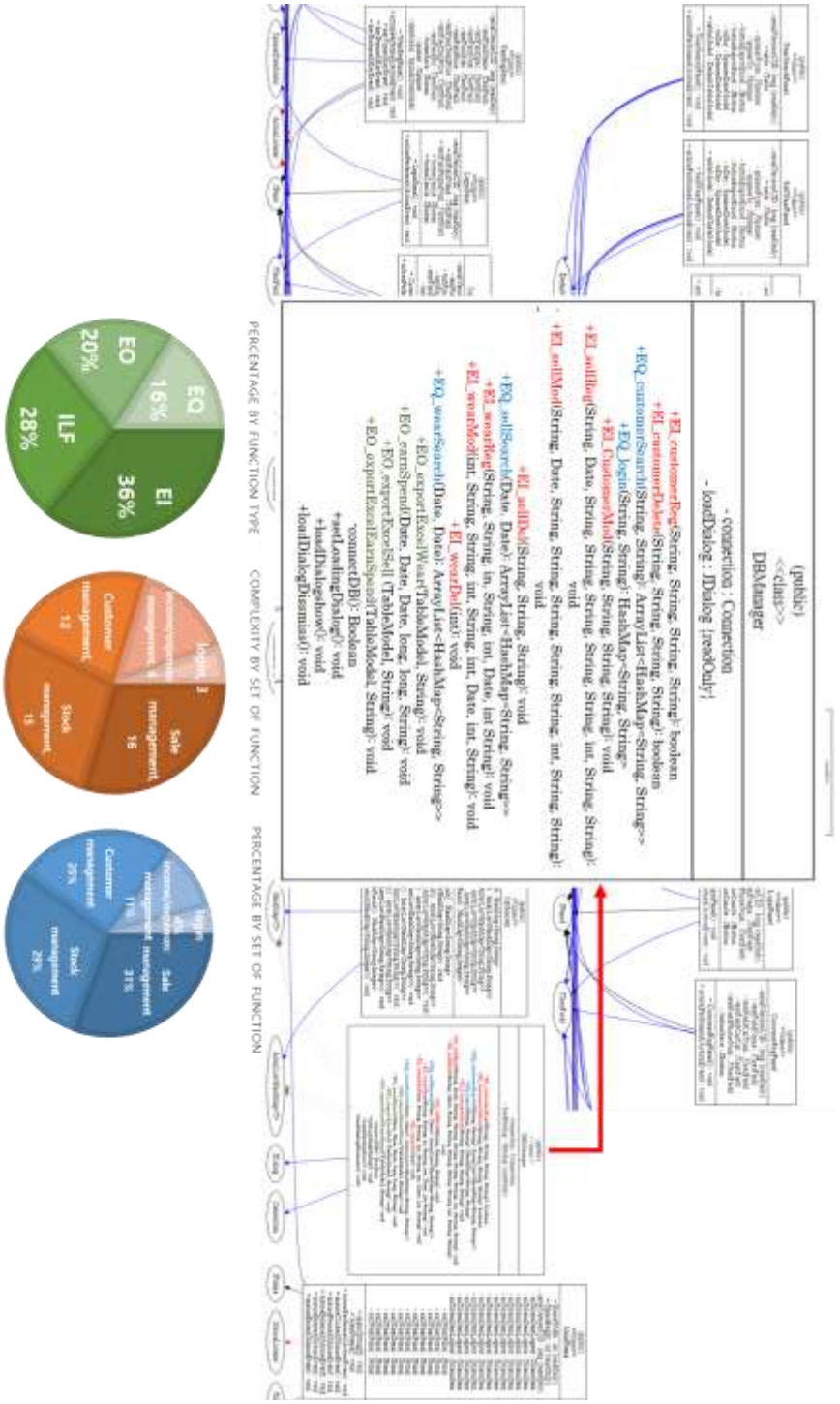

Figure. 6. Visualization for function point

Table 7. Function point of target program

No	Function Name	Transaction Type	FTR	DET	Complexity	Function Point	DET (Data Element Type)
Function 1	EI_customerMod	EI	1	4	Low	3	cartype, phonenumber, funcBtn1, message1
Function 2	EI_customerReg	EI	1	7	Low	3	customername, carnumber, cartype, phonenumber, funcBtn1, message1, message2
Function 3	EI_SellDel	EI	1	6	Low	3	customername, phonenumber, installname, installdate, funcBtn1, message1
Function 4	EI_sellMod	EI	1	10	Low	3	customername, phonenumber, installname, installdate, installcontent, installmodel, component, price, payment, master, funcBtn1, message1
Function 5	EI_SellReg	EI	1	10	Low	3	installname, installdate, installcontent, installmodel, component, price, payment, master, funcBtn1, message1
Function 6	EI_wearMod	EI	1	13	Low	3	salename, standard, quantity, unit, unitprice, totalprice, stockdate, saleprice, extra, funcBtn1, message1, func_calculation, func_number
Function 7	EI_wearReg	EI	1	13	Low	3	salename, standard, quantity, unit, unitprice, totalprice, stockdate, saleprice, extra, funcBtn1, message1, func_calculation, func_number
Function 8	EO_earnSpend	EO	2	10	Low	4	dateFrom, dateTo, date, exporttotalprice, func_calculation1, intotalprice, func_calculation2, revenue, func_calculation3, funcBtn1
Function 9	EO_exportExcelEarnSpend	EO	2	11	Low	4	dateFrom, dateTo, date, exporttotalprice, func_calculation1, intotalprice, func_calculation2, revenue, func_calculation3, funcBtn1, funcBtn2
Function 10	EO_exportExcelSell	EO	2	12	Low	4	customername, phonenumber, installname, installdate, installcontent, installmodel, component, price, payment, master, funcBtn1, message1
Function 11	EO_exportExcelWear	EO	1	11	Low	3	salename, standard, quantity, unit, unitprice, totalprice, stockdate, saleprice, extra, funcBtn1, message1
Function 12	EI_customerDelete	EI	1	4	Low	3	customername, carnumber, cartype, phonenumber, funcBtn1, message1
Function 13	EQ_customerSearch	EQ	2	9	Low	3	customername, carnumber, cartype, phonenumber, salename, price, date, funcBtn1, message1
Function 14	EQ_login	EQ	1	4	Low	3	customername, phonenumber, funcBtn1, message1
Function 15	EQ_SellSearch	EQ	1	12	Low	3	customername, phonenumber, installname, installdate, installcontent, installmodel, component, price, payment, master, funcBtn1, message1
Function 16	EI_wearDel	EI	1	11	Low	3	salename, standard, quantity, unit, unitprice, totalprice, stockdate, saleprice, extra, funcBtn1, message1
Function 17	EQ_wearSearch	EQ	1	12	Low	3	salename, standard, quantity, unit, unitprice, totalprice, stockdate, saleprice, extra, funcBtn1, dateFrom, dateTo

5. Conclusion

The cost or size estimation for a software development project is very important for the success of the project. An accurate size estimation allows manpower to be systematically managed and enables the causes of problems to be analyzed with the performances and plans of the developers. However, in a domestic project size estimation process, the client estimates the size instead of measuring the size, and the development company organizes the budget according to the size.

This study attempted to validate the project size estimation. The total function point for the data and transaction function was calculated by obtaining the function types from the source code through source code visualization based on reverse engineering and measuring the function point for each function. This study aimed to validate the size of a project at the time of ordering and at the time of completion. If verification is performed using the proposed method and the accumulated data are utilized for other project orders, it will be significantly helpful for the project size estimation. In the future, we intend to improve the current inconvenience of parsing by string and low accuracy by developing a parser capable of analyzing queries. In addition, the proposed method will be applied to real-world projects for comparison and verification, and data will be accumulated to improve the project performances and processes of organizations.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2017R1D1A3B03035421), and supported by Development of Information Communication and Broadcast R&D(2018 Open OS Environment Development and Diffusion) through NIPA(National IT Industry Promotion Agency)(S1113-18-1001), and supported by a grant(18CTAP-C133299-02) from Technology Advancement Research Program funded by Ministry of Land, Infrastructure and Transport of Korean government. Also, this work was supported by the Human Resource Training Program for Regional Innovation and Creativity through the Ministry of Education and National Research Foundation of Korea (NRF-2015H1C1A1035548).

References

[1]. Vidger, M.R., A. W. Kark, Software cost estimation and control, Institute for Information Technology, National Research Council Canada, 1994.

[2]. Low, G.C., D.R., Jeffery, “Function points in the estimation and evaluation of the software process”, IEEE Transactions on Software Engineering, Vol. 16, No.1, pp.64-71, 1990.

[3]. Kemerer, C.G., B.S., Porter, “Improving the reliability of function point measurement: An empirical study”, IEEE Transactions on Software Engineering, Vol. 18, No.11, pp.1011-1024, 1992.

[4]. Chan Gyu Park, Ja Hwan Gu, Seong Hee Kim, Soo Jeong Shin, Byeong Seon Song, “A Study on the Estimation of Software Development Cost of IT Projects in Public Sector”, Management Science, Vol. 19, No. 2, pp.191-204, 2002.11.

[5]. Yeon Shik Ahn, "An Enhanced Function Point Model for Software Size Estimation: Micro-FP Model", Korea Society of Computer Information, Vol. 14, No. 12, pp.225-232, 2009.

[6]. Roger S, Pressman, 2010, "Software Engineering: a practitioner's Approach (7th ed.)", McGraw Hill.

[7]. Ivan Sommerville, 2001, "Software Engineering (6th ed.)", Pearson Education.

[8]. So Young Moon, R. Young Chul Kim, "Code Structure Visualization with a Tool-Chain Method", Research India Publications, Vol. 10, No.90, pp.214-218, 2015.

[9]. Min Gyu Park, Eun Young Byun, Jeong Wha han, R. Young Chul Kim, So Young Moon, "Development of JDT Based Static Analyzer for Code Analysis", KIPS, Vol.22, No.2, pp.969-972, 2015.

[10]. So Young Moon, R. Young Chul Kim, Chae Yun Seo, "A Survey for Software Visualization", Journal of Platform Technology, Vol.4, No.3, pp.22-29, 2016.

[11]. So Young Moon, R. Young Chul Kim, "Validating the Cost Estimations of Function point through Code Visualization Mechanism based on Reverse engineering", KCSE, Vol.20, No.1, pp.204-205, 2018.

[12] Eun Young Byun, Hyun Seung Son, Byungkook Jeon, R. Young Chul Kim, "Reusability Strategy Based on Dynamic Reusability Object Oriented Metrics", Journal of Engineering Technology, Vol.6, Issue 1, pp.365-377, January 2018.

- Corresponding Author : Byungkook Jeon, R. Young Chul Kim