An Empirical Study on Adjustment Factors to Estimate Maintenance Cost of Applications Developed Using Components

Byoung-Chol Lee and Sung Yul Rhew

Abstract—**Recently, the efficiency of software development is dependent on the use of component modules and open source. However, these need the additional cost and effort for understanding and modification, which is basic process in software maintenance. In this research, adjustment factor which shows the maintenance cost of the black box embedded software like the component module and open source is proposed. In order to calculate the maintenance and repair adjustment factor, we have classified type of application system based on maintenance and repair specialized company K. And the methods for determine whether the adjustment factor applicable to the application is proposed. Also it confirmed that it need to adjust maintenance cost about 8% in result of an analysis of data and the discussion with the person in charge and experts about application type in which the component is included.**

Index Terms—**Software maintenance, maintenance cost, cost adjustment factor, maintenance cost estimation.**

I. Introduction

Software maintenance generally refers to any work done to change software so as to correct error occurring after the software system was delivered to a customer, to enhance the function or performance of software, or to allow system to have compatibility with changing environment. Software understanding is so important that it accounts for 50% of full efforts taken during software maintenance [1]. And commercialized software is increasingly used while developing software, so as to enhance software development productivity and quality and to shorten development time. That is an example of using components such as the user interface component, the communication module component, and the security module component.

Many companies are outsourcing software maintenance so as to reduce their internal costs and increase the utilization of latest technology [2]. In the early stage of introducing outsourcing into software, outsourcing used to be mostly confined to limited areas such as the development, and maintenance of system. Nowadays, however, outsourcing is expanding itself in the form of system integration, which incorporates system management and design, construction, operation, consulting, education, etc [3].

Like this, the scope of the application system maintenance is expanded and the complexity and the difficulty of it is also higher, which results in the increased burden of expenses on the part of maintenance service provider. Hence, this study proposes maintenance cost adjustment factors to remedy the limitations of function point-based maintenance cost estimation. It is expected that the application of maintenance cost adjustment factors to areas that cannot be estimated with function points will allow rational cost estimation.

II. Related Studies

In this chapter, existing studies on the estimation of system maintenance cost are investigated and their contents and limitations are looked into.

A. Software Scale Adjustment Factor

Software scale measurement models use adjustment factors to reflect the characteristics of software. The COCOMO II model uses as scale factors (SFs) 5 elements, which include Precedentedness (PREC), Development Flexibility (EFEX), Architecture/Risk Resolution (RESL), Team Cohesion (TEAM) required among members such as user, customer, and developer, Process Maturity (PMAT) levels in Capability Maturity Model Integration (CMMI) of a development organization. These elements are used in the scale estimation formula as variables in the form of exponents [4].

Value Adjustment Factor (VAF) of IFPUG (International Function Point User Group) is based on 14 general system characteristics (GSC), which rate the general functionality of applications measured. Each characteristic is expressed on the scale of 0 to 5. The purpose of VAT is to adjust the unadjusted function point count within ±35 % [5].

NESMA (Netherlands Software Metrics Association) defines three types of function point counts: detailed, estimated, and indicative. The detailed function point count does not apply General System Characteristics (GSC). The estimated function point count gives a default weight to each type of function [6].

COSMIC (Common Software Management International Consortium) proposed COSMIC-FFP for expanding the application scope of function point technique which is data-centered management in MIS(Management Information System) to embedded software and real-time system which based on control management. The COSMIC-FFP method, however, does not give weights to function which are related to scale estimation, though it gives descriptions of components of software function [7].

Manuscript received June 5, 2013; revised September 11, 2013.

The authors are with the SoongSil University, Seoul, 156-030, Republic of Korea (e-mail: wanarun@ naver.com, syrhew@ssu.ac.kr).

DOI: 10.7763/LNSE.2014.V2.101

B. Limitations of the Existing Studies

Currently, as for software size measurement, the functional size measurement method is used as a standard in Korean. However, there are still controversies over the accuracy and value of the method due to differences in users' subjective understanding in function measurement and its limits that it expresses the complexity of diverse functions with a simple fixed matrix.

To remedy such limitations, Boon Hee Byun [8] carried out complementary research on application type adjustment factors and language adjustment factors. In-Yong Jung et al. [9] proposed adjustment factors that reflect characteristics of organizations. Yeon S. Ahn [10] suggested a Micro-FP model, expanding twice the matrix of File Type Reference (FTR) and the Data Element Type (DET) so as to improve the current problem of complexity that the weights of transaction complexity are limited to low, average, and high.

Studies so far, however, have dealt with micro scale adjustment factors designed for the precision of software size measurement. This study proposes macro maintenance cost adjustment factors unlike the micro scale adjustment factors of the existing studies, so as to remedy limitations of the function point method.

III. Software Maintenance Cost Adjustment Factor

The function point estimation method judges functions from the user's viewpoint, and so in the case of a function that cannot be identified by a user, it fails to estimate its size. This chapter is going to discuss about a solution to this.

A. Areas Excluded from Cost Estimation

This study classifies application systems used by companies into three types as shown in Table I. Type A is an application system where software source codes can be identified. Type B is an application system developed with the partial use of a component, whose source codes cannot be identified. Type C is an application system that has customized a commercial product, and its source codes cannot be identified.

The function of a component is not apparent noticeably because it is included in application functions or is partly linked to them. Thus, there occur cases where the identification of some function is omitted from maintenance cost estimation. Application system Type B represents such cases. Therefore, this study proposes the maintenance cost adjustment factor for application system Type B.

TABLE I: Application System Type

NO	Type	White Area	Black Box Area
1	A	●	
2	B	●	●
3	C		●

B. Method to Judge Area Excluded from Cost Estimation

The maintenance cost adjustment factor proposed by this study does not apply to all application systems. Thus, this study set criteria items for deciding whether the adjustment factor may be applied or not.

The criteria items for decision were derived from the following procedure, using the delphi method. First, a review group was formed to select the criteria items. The review group consisted of a total of 34 persons, who included 19 persons in charge of maintenance, 7 maintenance managers, and 4 persons in charge of maintenance contract. A survey was conducted using open-ended questions so that the members of the review group might list according to priority criteria items suitable to be reflected in the adjustment factor. Ten most frequently mentioned items were selected among collected items, with one receiving higher priority being selected among ones with the same count, Table II.

TABLE II: The Decision Criteria to Apply the Adjustment Factor

No	Decision Criteria Items	Adoption
L1	A System developed using the COTS Component	◎
L2	A System developed using the Tool	
L3	A System customized the package software	◎
L4	A System Included Firmware Software	◎
L5	Using of API Function provided by OS	
L6	Interface with Other System	
L7	Including Network Communication Module	◎
L8	Including Security Module	◎
L9	The System focused on UI design	
L10	The System focused on Data Mart	

A list of the selected 10 items were prepared and sent to the members of the review group so that they might rate them according to the degree of influence on the period and efforts of maintenance. A 5-point Likert scale was used for their rating. And the force field analysis was used to represent the results of their rating, as in Fig. 1. Five items in Area 1 of Fig. 1 were selected as decision criteria for applying the adjustment factor. The selected items were marked in the field of adoption in Table II.

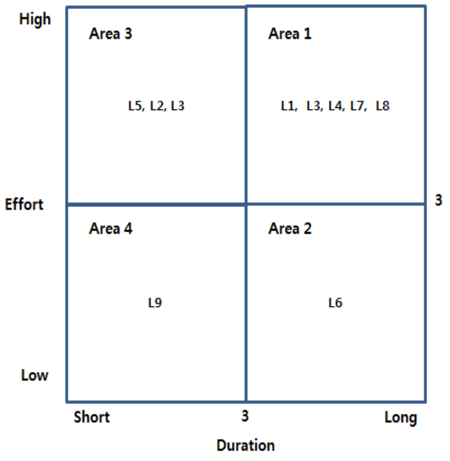

Fig. 1. Result of rating analysis.

C. Extraction of the Adjustment Factor

For extracting the adjustment factor, select a target application system, using the above decision criteria. Then, derive the adjustment factor, through analyzing an interview with maintenance personnel and the results of maintenance work. During an interview with the person in charge of operation, ask what type of external components has been applied and which activities were carried out for understanding the components. In addition, investigate what difference it has in terms of efforts and a period, compared to the understanding of a portion that has source code.

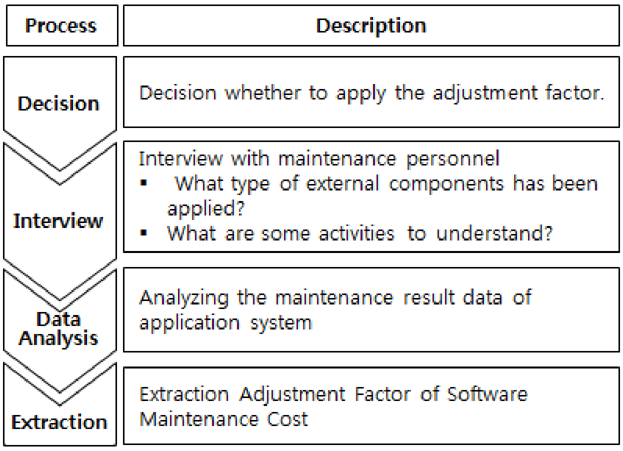
Process
Description
Decision
Decision whether to apply the adjustment factor.
Interview
Interview with maintenance personnel
▪ What type of external components has been applied?
▪ What are some activities to understand?
Data Analysis
Analyzing the maintenance result data of application system
Extraction
Extraction Adjustment Factor of Software Maintenance Cost

Fig. 2. Process of adjustment factor extraction.

In addition, so as to extract the adjustment factor in quantitative figures, analyze results of software maintenance. Use data that are burdensome for a maintenance service provider due to a long period of maintenance, rather than analyzing results of coping with all requests for maintenance. Among reasons for a long period of maintenance, target an application system that requires additional efforts for reasons of an external component, etc. Find the arithmetic mean of additional efforts needed for maintenance, and extract the adjustment factor. Fig. 2 explains a procedure for the extraction of the adjustment factor. Use the extracted adjustment factor for the estimation of additional software maintenance cost.

IV. Case Study

This chapter confirms the application system maintenance scale adjustment factor proposed through the case study of application maintenance performed by a maintenance service provider.

A. A Case of Company K's Maintenance

A case of Company K is used to extract the maintenance cost adjustment factor proposed by this study. Company K, which has performed outsourcing service for over 20 years, is now entrusted with software maintenance by eight companies. This study used for case data Company S' application system, one of Company K's clients.

TABLE III: Summary of Application System Maintenance

NO	Type	Numbers	Man day	Average Man day
1	A	25	820	32.8
2	B	19	695	36.57
3	C	2	43	20.5

The data of analysis were confined to maintenance carried out from 2012 to April 2013. During the period of analysis, there were a total of 452 requests for pure maintenance such as the addition or modification of application system functions. Among them, 46 cases for which over 20 days were spent for maintenance work were rearranged, as shown in Table III, according to the whole period of maintenance and the average days of completion.

B. Maintenance Cost Adjustment Factor

For estimating the maintenance cost adjustment factor on the basis of results of maintenance performed, interviews were conducted with maintenance personnel so as to find out reasons for prolonged periods of maintenance. The reasons ranged from system adaptation to change in business regulations or legal system to requests for adding many functions, understanding of external components or packages used, and the like.

Since this study was to apply the adjustment factor to omitted portions from cost estimation, data were analyzed and the adjustment factor was extracted only with respect to Type B as shown in Table IV. As for Type A, functions were identifiable, and as for Type C, most maintenance was entrusted to the manufacturer; and so it was deemed that they had no portion neglected in function point estimation. Therefore, the adjustment factor was not estimated as to application system of Type A and Type B. The professionalism of a person in charge of function-point estimation was left out of the discussion. As for application system of Type B, it was found that additional efforts of 13% was needed in maintenance.

TABLE IV: Additional Maintenance Effort by Application System Type

No.	Type	Additional Effort
1	A	13%
2	B	
3	C	

C. Software Maintenance Cost

The software maintenance cost of Company S carried out by Company K was estimated again in accordance with the method proposed by this study. The recalculation was conducted, as shown in (1), reflecting the adjustment factor extracted above into the total cost of maintenance before adjustment.

Formula : Calculation of Application System Maintenance Cost

$$\begin{aligned}&\textit{Total Cost of Application System Maintenance}\\&=(FC+VC)\times HRISM\end{aligned} \quad (1)$$

FC=Fixed Cost
VC=Variable Cost
HRISM=Hidden Rate of Application System Maintenance

Table V shows the results of applying the adjustment factor after classifying Company S' application systems. At present, Company K receives from Company S maintenance cost at the rate of 10% of development cost for a total of 49 business systems, among which 16 business systems fall under those to which the maintenance cost adjustment factor is applicable. The results of recalculating maintenance cost with the application of the maintenance cost adjustment factor show an effect of 8% increase in cost compared to the previous total maintenance cost.

TABLE V: SUMMARY OF APPLICATION SYSTEM MAINTENANCE

NO	Type	Numbers	Additional Rate of Maintenance Cost
1	A	18	13%
2	B	16	
3	C	25	
4	Total (A + B + C)	49	**8%**

V. CONCLUSION AND FUTURE STUDY

Although the use of commercial components is increasing in the development of software in order to enhance development productivity or improve quality, some major areas are found to be omitted from function cost measurement. Therefore, this study proposed cost adjustment factors to overcome the limitations inherent in the maintenance cost estimation using function points. The adjustment factor proposed by this study is a macro maintenance cost adjustment factor unlike those proposed by the existing studies. This study, however, only shows a direction within the big framework of maintenance cost adjustment factors, and has limitations in accuracy of the adjustment factors.

Hence, further studies are needed for the completedness of the maintenance cost adjustment factor proposed herein. The types of application systems are needed to be subdivided more detailedly. And it is necessary to study methods for measuring, with function points, the subdivided application areas that cannot be measured with function points. Further, additional studies are needed with respect to factors that have effects on the adjustment factor.

REFERENCES

[1] R. S. Banker, S. Datar, and C. Kmerer, “A model to evaluate variables impacting the productivity of software maintenance project,” *Management Science*, vol. 37, no. 1, pp. 1-18, Jan. 1991.

[2] V. Grover, M. J. Cheon, and J. T. C. Teng, *Journal of Management Information Systems*, vol. 12, no. 4, pp. 89-116, 1996.

[3] N. KiChan and L. Jae-Nam, *Information System Outsourcing: Methodology and Examples*, Ajin, pp. 37-40, 1999.

[4] B. Boehm, B. Clark, E. Horowitz, and C. Westland, “Cost models for future software life cycle processes: cocomo 2.0,” *Annals of software engineering*, 1995, vol. 1, no. 1, pp. 57-94.

[5] IFPUG, *Function Point Count practices Manual, R4.2.1*, Internation Function Point Users Group, 2006.

[6] NESMA, *Definition and Counting Guidelines for the Application of Function Point Analysis: A Practical Manual. version 2.2*, Netherlands Software Measurement user Association, 2004

[7] N. Monaka, A. Kakural, E. Bukhary, and M. Azuma, *A Complexity-Weighted Functional Size Metric for Interactive Software*, Advanced Institute for Science & Eng., Waseda University, 2002.

[8] B. H. Byun and K. T. Kwon, “A study on the derivation and sensitivity analysis of the adjustment factor in the software cost estimation guidelines,” *KIPS* Software *Engineering*, 15-D, no. 1, pp. 61-72, Feb. 2008.

[9] I. Y. Jung, D. J. Woo, J. H. Park, and C. S. Jeong, “Improved function point measurement model for software size estimation,” *Journal of Korean Society for Internet Information*, vol. 10, no. 4, pp. 115-126, Aug. 2009.

[10] Y. S. Ahn, “An enhanced function point model for software size estimation: micro-fp model,” *Journal of KSCI*, vol. 14, no. 12, pp. 225-232, Dec. 2009.

Byoung Chol Lee received the MS and BS degrees in computer science from Soongsil University, Seoul, Korea, in 2008 and 1996, respectively. He is Ph.D Candidate at SoongSil University, Korea. He is working as a deputy manager at KEPCO KDN, Korea. His current research interests include software maintenance, internet ethics.

Sung Yul Rhew received the Ph.D degree in computer science from Ajou University and the MS degrees in computer science from Yonsei University , Seoul, Korea, in 1996 and 1980, respectively. He is working as a Professor of software engineering in school of computer science and engineering at the Soongsil University. His research interests include software requirement engineering, software maintenance, open source software.