International Journal of Applied Engineering Research, ISSN 0973-4562 Vol. 10 No.90 (2015)

Extending Reverse Engineering for Software Maintenance

Woo-Sung Jang [1], R.Young Chul Kim[2*], Jae-Hyup Lee[3]
[1, 2*]SE Lab, Hong-ik University, Sejong, Korea
{jang, bob}@selab.hongik.ac.kr
[3]Dept. of Computer Engineering, Koreatech University, Korea
jae@koreatech.ac.kr

Abstract - Software Maintenance is an important portion in the cost of an entire software development. However, it is not possible to prepare with protecting all risks at the stage of design. Applying maintainability with ISO/IEC 9126 can be improved. But in the case of complex software, it will take a lot of time to find the location of the risk elements for improvement. To solve this problem, we propose an improved solution based on reverse engineering. First, calculates the scoring of maintainability through expanding the calculation method of ISO/IEC 9126 maintainability. Second, lists the priority of the location to require an improvement. And also enhances the user understanding with a whole class-related diagram, that is, displaying the entire system based on layer architecture.

Keywords - Maintainability; Reverse Engineering; Layer Architecture; ISO/IEC 9126

I. INTRODUCTION

Software maintainability and modifiability are important the quality characteristics of software. The quality characteristics are known to occupy a major portion of the cost of software development life-cycle (SDLC). Therefore, maintainability of a software system can have a great effect on software cost [1]. According to Fred Brooks‟ Software Engineering, maintenance cost for a typical program occupies more than 40% of the total cost. Hewlett-Packard (HP) mentioned that about 60~80% of R&D personnel participates in maintenance activities, and maintenance cost constitutes about 40~60% of production cost [2]. In order to reduce this maintenance cost, a thorough design is required. However, it is not possible to anticipate to deal with all risks at design stage. In modifications to maintain completed software involves an unexpected risk.

ISO/IEC 9126 defines characteristics with product quality of software, and presents objective methods to measure its characteristics [3]. As maintainability is one of the characteristics of ISO/IEC 9126, it can improve with the use of ISO/IEC 9126. However, in the case of complex software, it takes a lot of time to locate a part to require maintenance.

In this paper, we propose a solution for a software maintainability improvement based on reverse engineering. First, scores for maintainability, PMD, LOC, and Coupling extracted from a completed software source code using the Score Tool Chain. Extracted scores are outputted on Dash Board. A connection diagram is drawn of modules in software. The diagram can be expanded or reduced at component (class), package, or system level. And in order to present the part to require an improvement preferentially according to the maintenance goal of the organization, the eight values are added to 4 sub-characteristics of maintainability. The user can identify the location to require an improvement on Dash Board as well as the corresponding class & connected classes. In this case, the work period is limited, it can be enhanced by improving the function that is close to the maintenance goal of the project preferentially.

This paper is as follows: As a related study, Chapter 2 discusses measurement methods of maintainability scores on ISO/IEC 9126, and classification methods for quality attributes of Clarrus Consulting Group Software. Chapter 3 discusses on software maintainability improvement solution based on reverse engineering. Chapter 4 discusses on an application examples using our proposed methods. Lastly, mentions the conclusion and future research.

II. RELATED WORKS

2.1 Measurement Methods of ISO/IEC 9126 Based on Maintainability Scores

ISO/IEC 9126, an international standard, defines characteristics of software quality and metrics of quality assessment, and provides an explanation from the perspective of a user or a developer. These characteristics and sub-characteristics are shown in Figure 1. Figure 1 mentions characteristics and sub-characteristics of ISO/IEC 9126 [4].

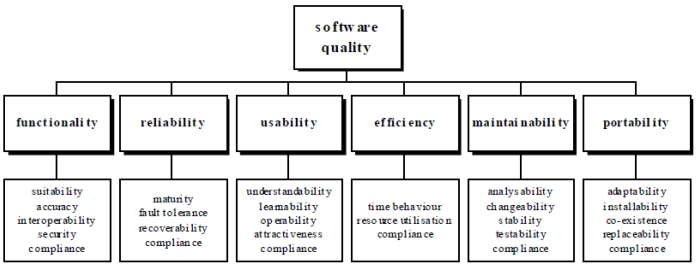

Figure 1. Characteristics and sub-characteristics of ISO/IEC 9126 [4]

Defense Agency for Technology and Quality has created metrics used to define quality standard of software based on maintainability characteristics of ISO/IEC 9126. In order to

quantify the result of source code analysis, and to support the result visibly, these formulas have been defined to generate source code based on scores [5]. More detailed explanations are provided as shown in Table 1. Table 1 shows Maintainability Sub-characteristics Formulas.

Table 1. Maintainability Sub-characteristics Formulas [5]

Sub-characteristics	Formula	ETC
Analyzability	WMC X 25 + STMT X 25 + DIT X 25 + CD X 25	WMC (Weighted Methods per Class) STMT (Number of Statements) DIT (Depth of Inheritance) CD (Comment Density)
Testability	RFC X 25 + CBO X 25 + DIT X 25 + NOM X 25	RFC (Response for a Class) CBO (Coupling Between Object) DIT (Depth of Inheritance) NOM (Number Of local Method)
Stability	WMC X 25 + LCOM X 25 + CBO X 25 + DIT X 25	WMC (Weighted Methods per Class) LCOM (Lack of Cohesion Of Methods) CBO (Coupling Between Object) DIT (Depth of Inheritance)
Changeability	WMC X 25 + RFC X 25 + SIX X 25 + PubMR X 25	WMC (Weighted Methods per Class) RFC (Response for a Class) SIX (Specialization Index (%)) PubMR (Public Methods Ratio (%))

2.2 Selective Clustering Diagram System for Software Architecture

Selective Clustering Diagram System for Software Architecture is a Toolchain to draw module connection diagrams through an analysis of source codes based on reverse engineering. We can extract architecture possible to component (class), package, or system level for manually selection of modules per each level. And we can automatically draw a graph with DOT library [7]. The entire structure is shown below in Figure 2. Figure 2 shows the tool chain to generate the architecture graph based on reverse engineering.

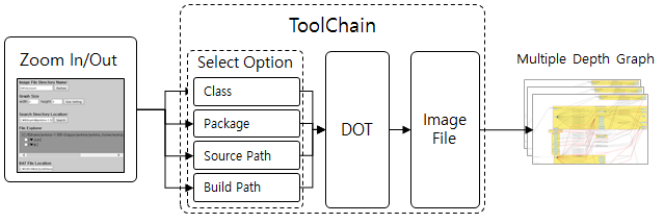

Figure 2. The Toolchain for extracting the architecture graph

Figure 3 shows actual implementation of Selective Clustering Diagram System for extracting Software Architecture. On the right side of figure 3, an input window displays to enter the user"s selection, whereas it displays to generate a graph on the left side of figure 3.

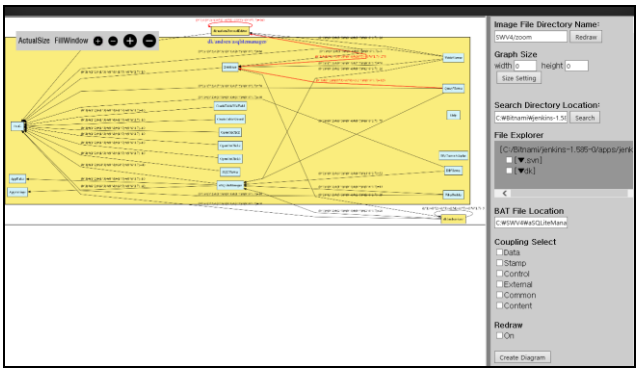

Figure 3. Selective Clustering Diagram System for extracting Software Architecture

III. SOFTWARE MAINTAINABILITY IMPROVEMENT SOLUTION BASED REVERSE ENGINEERING

We propose a method to extract design from software source codes based on reverse engineering techniques, and to provide indexes to maintain the source codes. The method extracts maintainability scores from codes with Score Toolchain to display them on Dash Board, and also to show the visual graph of the modules within the source codes.

The graphs can be expanded or reduced with component (class), package, or system level respectively on Graph Toolchain for Selective Clustering Diagram System for extracting Software Architecture. When the user selects class level, package level, or system level, it automatically draws a desired component (class, package, or system) with a connected diagram. In the diagram, the coupling scores on each component are displayed.

The user can identify classes to fix on Dash Board, and refactor the source codes for the improvement. Then after rebuilding the improved source codes, it re-shows the improved maintainability score. Figure 4 shows Software maintainability improvement mechanism based reverse

emgineering.

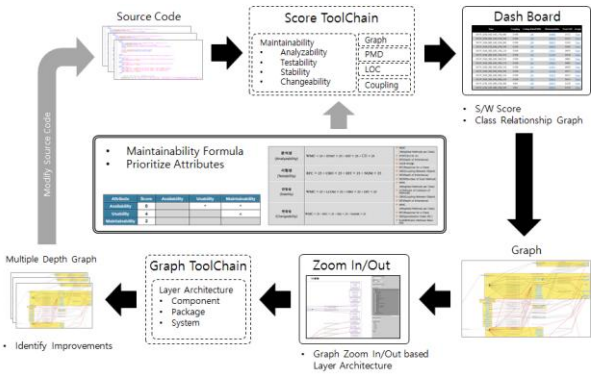

Figure 4. Software maintainability improvement mechanism based reverse emgineering

3.1 The Calculation Method of the Score Toolchain's Maintainability Score

The maintainability score of the Score Toolchain is calculated on using Prioritization of Selected Quality Attributes[6], which assigns the priority weight value applied at the second stage of the classification method of quality attributes of Clarrus Consulting Group software. We use the weight value for modifying source code in prior.

In order to calculate the formulas in Prioritization of Selected Quality Attributes[6], we need to sum 10 attribute values. The 10 attribute values can be extracted through parsing of source codes. As for a tool for extracting attribute values, there exist quite a few open source libraries. But for attributes not supported by libraries, we use a self-created algorithm to extract the values for source code analysis.

Weight values for 4 sub-characteristics of maintainability are differently assigned depending on the property of a particular project. Table 2 shows weight values of a project for changeability-oriented maintenance. In order to multiply each formula by weight value, it adds „1" to each score.

Table 2. Prioritization of Maintainability Sub-characteristics based on [6]

	Score	Analyzability	Changeability	Stability	Testability
Analyzability	2		>	>	<
Changeability	4			<	<
Stability	3				<
Testability	1				

Maintainability score is calculated through multiplication of sub-characteristics by corresponding weight value. The formula is shown in Table 3. Table 3 shows the maintainability formula with added weight values.

Table 3. Maintainability Formula with Added Weight Values

Attribute	Formula
Maintainability	Analyzability x 2 + Changeability x 4 + Stability x 3 + Testability x 1

3.2 The Score Toolchain

In addition to maintainability score, the Score Toolchain extracts scores of PMD, LOC (Line of Code), and Coupling as well as class connection graphs from the source codes inputted by the user. The extracted data are stored in XML, and image files. The stored files can be viewed on Dash Board. Figure 5 shows the detailed structure of the Score Toolchain.

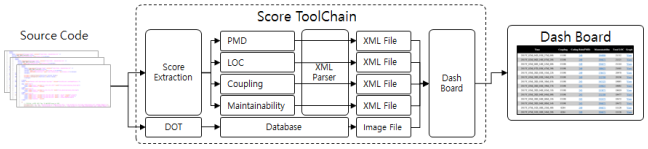

Figure 5. The detained structure of the Score Toolchain

Other scores on DOT are listed as follows:

- PMD: A static rule-set based Java source code analyzer to identify potential problems. It has functions inspecting for possible bugs, dead code, overcomplicated expressions, suboptimal code, and duplicate codes. The inspected results are represented in score, provided by open libraries [8].
- LOC: Line of Code, that is, the total number of code lines used in the project.
- Coupling: Coupling relationship between classes, existed by diverse open libraries.
- DOT: Graph generation by matching the rule-set database with source codes.

It displays the calculated scores and graphs on Dash Board. Table 4 shows the structure of Dash Board. The time on the Dash Board refers to the date built by the Toolchain. Coupling, PMD, and LOC refer to the scores calculated by respective libraries. It also displays maintainability graphs with detailed maintainability score at the implementation stage.

Table 4. Structure of Dash Board

Time	Coupling	PMD	Maintainability	LOC	Graph
2015-01-01	100	300	114500	21107	Link
…	…	…	…	…	…

IV. CASE STUDY

We show our implementation of the software maintainability improvement solution based on reverse engineering in Figure 6, 7. Source codes are uploaded to Jenkins solution. During building, it executed by the Score ToolChain. Score data and graphs are inputted to Dash Board. The user can identify maintainability score at the Dash Board. Also, through Zoom In/Out, it is clearly identified their relationship between classes. When completed improvement of source codes, it can be built again through Jenkins and be identified the scores of improved source codes on Dash Board.

The retrieved data in XML files on Dash Board are shown in Figure 6. When selecting values for each PMD and Maintainability, it is possible to see detailed information. When choosing a view link of Graph, it is possible to view graphs.

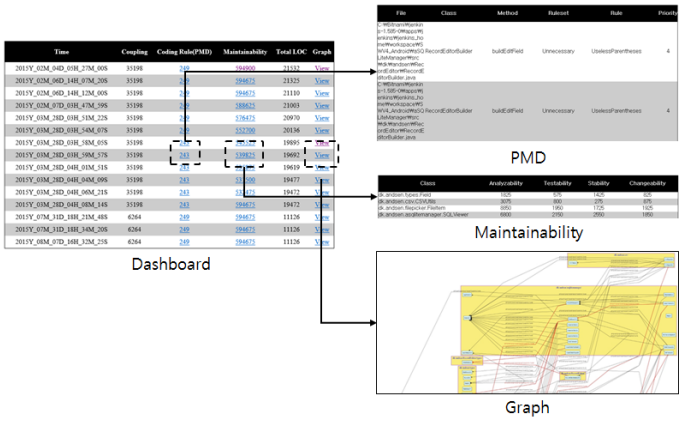

Figure 6. Implementation of Dash Board

Figure 7 shows the identified relationship between particular classes through Zoom In/Out. It is possible to display that only a part of the entire software structure is outputted.

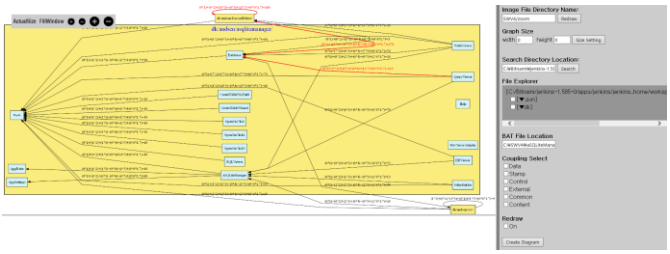

Figure 7. Expanded Class Relationships

V. CONCLUSIONS

This paper enhances maintainability of completed software and understanding of structure of the entire system to the developer or the manager. For the purpose, we suggest a solution based on reverse engineering to improve software maintainability. It is extracted scores of Maintainability, PMD, LOC, and Coupling from source codes with the Score Toolchain. For Maintainability and PMD, scores per class can be identified. That is, we propose an improved solution based on reverse engineering. First, calculates the scoring of maintainability through expanding the calculation method of ISO/IEC 9126 maintainability. Second, lists the priority of the location to require an improvement. And also enhances the user understanding with a whole class-related diagram, that is, displaying the entire system based on layer architecture.

In the future, research will be carried out to represent information on Dash Board into Graph, so that they could be applied to diverse examples.

ACKNOWLEDGMENT

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) and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2011601).

REFERENCES

[1] Mehwish Riaz, Emilia Mendes, Ewan Tempero., 2009, “A Systematic Review of Software Maintainability Prediction and Metrics,” ESEM '09 Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, pp. 367-377

[2] Don Coleman, Dan Ash, Hewlett-Packard, Bruce Lowther, Micron Semiconductor, Paul Oman., 1994, “Using Metrics to Ecaluate Software System Maintainability,” IEEE Computer, 27(8), pp. 44–49

[3] Yiannis Kanellopoulos, Ilja Heitlager, Christos Tjortjis, Joost Visser., 2008, “Interpretation of Source Code Clusters in Terms of the ISO/IEC-9126 Maintainability Characteristics,” CSMR '08 Proceedings of the 2008 12th European Conference on Software Maintenance and Reengineering, pp. 63-72

[4] ISO., 2003, “ISO/IEC TR 9126-2: Software engineering - product quality - part 2: External metrics,” Geneva, Switzerland.

[5] Defense Agency for Technology and Quality., 2012, “Weapons SW Quality Measurement Study”

[6] Clarrus Consulting Group Inc., 2010, “Software Quality Attributes: Following All the Steps”

[7] Graphviz., “http://www.graphviz.org”

[8] PMD., “http://pmd.github.io”

[9] Jenkins., “http://jenkins-ci.org”