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.

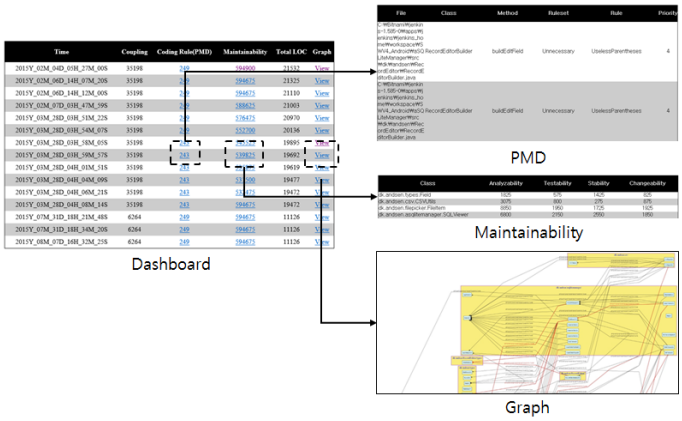

Figure 6. Implementation of Dash Board

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.

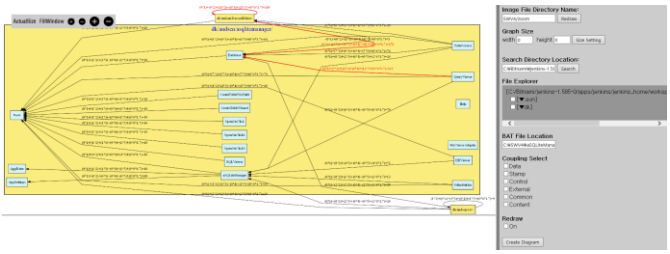

Figure 7. Expanded Class Relationships

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”