*Transactions of the Korean Nuclear Society Autumn Meeting*
*PyeongChang, Korea, October 25-26, 2007*

# An Introduction to Quantitative Measures for Software Maintenance of Nuclear Power Plant

Hyun Jun Jo and Poong Hyun Seong
*Department of Nuclear & Quantum Engineering, Korea Advanced Institute of Science and Technology*
*camp2006@kaist.ac.kr*

## 1. Introduction

The I&C system of NPP has changed from the analog system to the digital-based system using microcontrollers and software. Thus, software has become very important for NPP control system. The software life cycle is divided into the development and maintenance phase largely. Because poor software maintenance work introduces new errors and makes software much complex, we have to consider the effective maintenance methods for the reliability and maintainability of NPP software.

Function Block Diagram (FBD) is a standard application programming language for the Programmable Logic Controller (PLC) and currently being used in the development of a fully-digitalized reactor protection system (RPS) under the KNICS project. Therefore, the maintenance work will be of great importance in a few years. This paper studies on the measures which give quantitative information to software maintainer and manager before and after modification.

The remainder of this paper is organized as follows. Section 2 briefly describes software maintenance types and model. In Section 3-5, we introduce the quantitative measures for software maintenance and characteristics of FBD program. A conclusion is provided in Section 6.

## 2. Software Maintenance

Software maintenance means changes that have been made to computer programs after they have been developed. Software maintenances are performed to correct errors, implement changes, and enhance existing requirements[1]. It is important that the reliability and maintainability of modified software be maintained or enhanced after a change especially in safety-critical and long-life software.

### *2.1 Maintenance Types*

Software maintenance was classified by Swans into three types. First, corrective maintenance is to identify and correct software failure. Second, adaptive maintenance is to adapt software to changes in the requirements or environments. Finally, perfective maintenance is to enhance performance or improve maintainability. Corrective maintenance is dominant in hardware, but most software maintenance is adaptive and perfective maintenance because of software characteristics[1].

### *2.2 Maintenance Models*

There are a few models for software maintenance. Recent software maintenance models have included impact analysis and accounted for ripple effect as one of their stages[2]. The Pfleeger and Bohner model, SADT(Structured Analysis and Design Technique) Diagram of Software Maintenance Activities[5](see Figure 1) has six phases including change impact and ripple effect analysis.

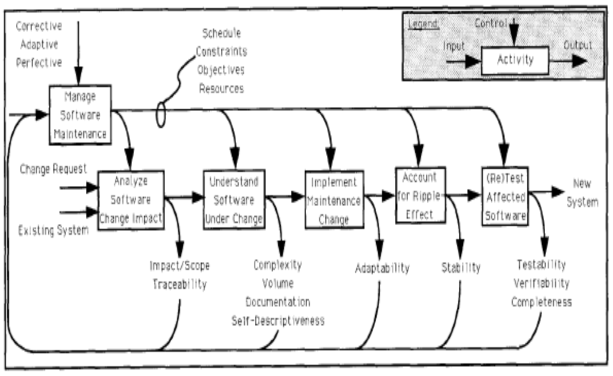

Figure 1. SADT Diagram of Software Maintenance Activities[5]

## 3. Change Impact Analysis

Change impact analysis (CHI) is to evaluate the effects of a proposed change. CHI has been defined as "identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change"[6]. Waiting until the change is made is far too late in the maintenance process to evaluate the effects of the change. Requirement traceability is a method of examining change before its implementation.

### *3.1 Traceability graph*

It describes the relationships within a workproduct and between workproducts(vertical and horizontal traceability)[5]. It helps the maintainer overview the products related with a change. Thus the impact of changes can be identified and checked during the modification of a program.

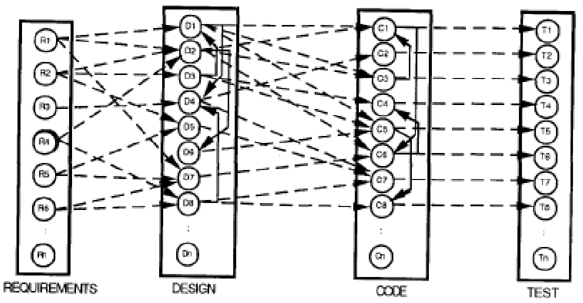

Figure 2. Traceability Graph[5]

### *3.2 Quantification*

The change impact can be assessed using traceability metrics. Measures such as cyclomatic complexity can be applied to the graph to determine if the overall system is likely to become more complex if the proposed change is made[5]. This measure help maintainer and manager to choose the design alternatives.

## 4. Ripple Effect Analysis

Ripple effect analysis(REA) is to evaluate the propagation of changes to other code modules as a result of the change just implemented[5]. Ripple effect can show the maintainer how great the effect of a change will be on the rest of the program or system. It can highlight modules with high ripple effect as possible problem modules[2].

### *4.1 Performance and Logical REA*

REA is divided into performance and logical REA. When several program variables are redefined in order to introduce an acceptable solution for a change, logical REA is accomplished using error-flow analysis. Performance ripple effect analysis requires the identification of modules whose performance may change as a consequence of software modifications. Performance dependencies often exists among modules which are otherwise functionally and logically independent [3].

### *4.2 Logical ripple effect and stability measure*

The stability of a program has been defined as the resistance to the potential ripple effect that the program would have when it is modified. Logical ripple effect is computed based on the impact of modifications using error flow[4]. A figure-of-merit is then proposed to estimate the complexity of program modification. It can highlight modules with high ripple effect as possible program modules and help the maintainer compare modifications[2].

## 5. FBD PLC Program

There are a number of programming languages used in PLCs. The IEC 61131-3 standards include five : Structured Text(ST), Function Block Diagram(FBD), Ladder Diagram(LD), Instruction List(IL) and Sequential Function Chart(SFC). The FBD is one of the most widely used languages because of its graphical notations and because it is useful in applications involving information or data flow between control components that can be designed as a network of software blocks.

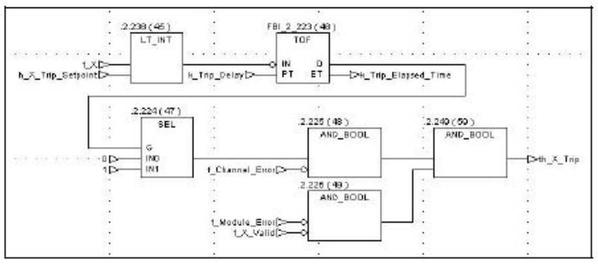

Figure 3. A example of Function Block Diagram

## 6. Conclusion

It is important that the reliability and maintainability of software be maintained or enhanced in the NPP software maintenance. Therefore the effective maintenance methods are considered for the reliability and maintainability. The traceability metrics using cyclomatic complexity is useful for choosing the design alternatives. Especially, REA can be used in an industrial environment for developing safety-critical applications for ensuring system and software reliability. There are few researches of CIA and REA in the PLC program maintenance. Thus, future work will focus on doing CIA and REA measures for FBD PLC program and proposing methods to reduce CI and RE.

## REFERENCES

[1] James Martin & Carma Mcclure, “Software Maintenance : The Problem and Its Solutions”, Prentice-Hall, Chapter 2 & 17.
[2] Sue Black, Computing ripple effect for software maintenance, Journal of Software Maintenance and Evolution, Vol.13, pp.263-279, 2001.
[3] S. S. Yau, J.S. Collofello and T. MacGregor, RIPPLE EFFECT ANALYSIS OF SOFTWARE MAINTENANCE, IEEE, 1978.
[4] S. S. Yau, J.S. Collofello, SOME STABILITY MEASURES FOR SOFTWARE MAINTENANCE, IEEE, 1979.
[5] Shari Lawrence Pfleeger and Shawn A. Bohner, A Framework for Software Maintenance Metrics, IEEE, 1990.
[6] Jacqueline Hewitt, Juergen Rilling, A Light-Weight Proactive Software Change Impact Analysis Using Use Case Maps, IEEE International Workshop on Software Evolvability, 2005.
[7] W. T. Tsai, R. Mojdehbakhsh, F. Zhu, Ensuring System and Software Reliability in Safety-Critical Systems, Department of Computer Science and Engineering, University of Minnesota.