Transactions of the Korean Nuclear Society Spring Meeting
Gwangju, Korea, May 30-31, 2013

A Method to Select Test Input Cases for Safety-critical Software

Hee Eun Kim[a], Han Seong Son[b], Hyun Gook Kang[a,]*
[a] Department of Nuclear and Quantum Engineering, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong,Yuseong-gu, Daejeon 305-701, South Korea
[b] Department of Game Engineering, Joongbu University, 201 Daehak-ro, Chubu-myeon, Geumsan-gun, Chungnam 312-702, South Korea,
**Corresponding author: hyungook@kaist.ac.kr*

1. Introduction

Software reliability is generally accepted as the key factor in software quality since it quantifies software failures which can make a powerful system inoperative [1]. In the KNICS (Korea Nuclear Instrumentation & Control Systems) project, the software for the fully-digitalized reactor protection system (RPS) was developed under a strict procedure including unit testing and coverage measurement [2]. Black box testing is one type of Verification and validation (V&V), in which given input values are entered and the resulting output values are compared against the expected output values [3]. Programmable logic controllers (PLCs) were used in implementing critical systems and function block diagram (FBD) is a commonly used implementation language for PLC [4]. This paper proposes a new testing methodology for effective and realistic quantification of RPS software failure probability.

2. FBD Testing

The test inputs for the RPS software should reflect the characteristics of safety-critical software. Additionally, the number of test cases needs to be reduced for practical reasons, although the test still needs to be exhaustive.

2.1 FBD and FBD Testing

The main characteristic of the PLC programs is indefinite and cyclic execution. Program reads input, computes new internal states, and updates output in each scan cycle [4]. The input for RPS software is process parameters that determine the trip, and the output is the comparison result of the input and the trip set point (TSP). Therefore, the test case for the RPS software consists of process parameters, statuses of modules and the trip signals.

2.2 Problems on the Software Testing

One of the problems on the software testing is the reflection of the past input sequences. Software inputs affect the program in some way, so the testing needs to reflect this effect. However, if the test case includes the past input sequences, the number of test cases explosively increases because it increases by 2^k for one k-bit digital input.

Another problem is reflection of software aspect. An aspect of a program is a kind of environment of a running program. The environment of a program may affect the function of a program, but it is hard to define and quantitatively evaluate its effect on the program.

3. Proposed Method

A unit FBD program consists of function blocks necessary to compute a primary output. The primary output, stored in the PLC memory, becomes an external output or used internally as input to other FBD units [4]. Therefore, the external inputs, internal state, and the aspect of software need to be reflected to evaluate software reliability. This is shown in the Fig. 1.

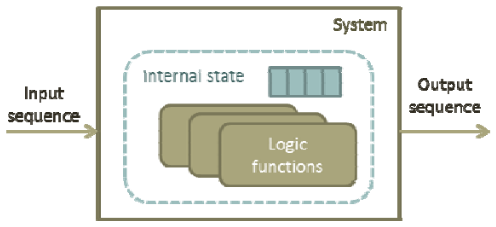

Fig. 1. Conceptual diagram of RPS software

3.1 Inputs

Previous study [5] introduced a testing method for software failure probability estimation. This method is developed based on the plant dynamics, characteristics of digital systems, the limited ADC resolution and the scan time. By applying this method, possible input values for the digital systems are tested, reducing number of test cases.

3.2 Internal State

Internal state refers to a state of a running program that may influence the behavior of the output of a program. Internal state is determined by the past input sequence and it can be represented by a set of internal variables. The data of the previous scan stored in the memory, which is internal variables, should be loaded when needed.

Therefore, by adding internal variables to the test case for the RPS software, we can reflect past input sequences on the software testing. The test case should include some input sequences to make specific internal state, or specific value of internal variables.

3.3 Aspect

As shown in the Fig. 2, the environment of software influences on its applications via the OS of the system. Therefore, the analysis on the causes of failure related to the aspect is less important in this study. In this case, the mean time between failures can be reflected on the software failure probability quantification.

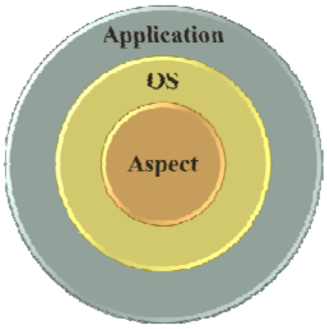

Fig. 2. Structure of software executed in real environment

However, aspect is not reflected to the test case generation step, but to the failure probability quantification step. (Fig. 3)

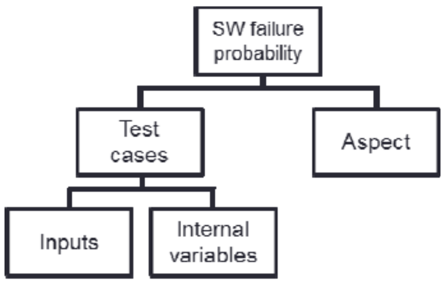

Fig.3 Schematic diagram of software failure probability quantification method

4. Conclusions

Software failure probability quantification is important factor in digital system safety assessment. In this study, the method for software test case generation is briefly described. The test cases generated by this method reflect the characteristics of safety-critical software and past inputs. Furthermore, the number of test cases can be reduced, but it is possible to perform exhaustive test. Aspect of software also can be reflected as failure data, so the final failure data can include the failure of software itself and external influences.

REFERENCES

[1] Michael R. Lyu, Handbook of Software Reliability Engineering, McGraw-Hill, 1996
[2] J. H. Park, D. Y. Lee, C. H. Kim, Development of KNICS RPS Prototype, Proceedings of ISOFIC Probabilistic Safety Applications 237 2005, Session 6, pp.160-161, Tongyeong, Korea
[3] P. H. Sung, Reliability and Risk Issues in Large Scale Safety-critical Digital Control Systems, Springer, 2008
[4] Eunkyoung Jee, A data flow-based structural testing technique for FBD programs, Information and Software Technology, Vol. 51 (2009), 1131–1139
[5] H. G. Kang, H. G. Lim, H. J. Lee, M. C. Kim, S. C. Jang , Input-profile-based software failure probability quantification for safety signal generation systems, Reliability Engineering and System Safety, Vol. 94 (2009) 1542–1546