A Profile-based Method to Select Test 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

Nowadays most of instrumentation and control (I&C) systems in the nuclear power plant (NPP) are digitalized and KNIC RPS is one of the fully-digitalized reactor protection systems. The software in the digital system enables the system to perform several safety functions. Thus the software in the KNICS RPS is crucial to the safety of a nuclear power plant in that its malfunction may result in irreversible consequences [1].

There were several studies to analysis the reliability of the RPS software, for example, software fault tree analysis [1] and input-profile-based software failure probability quantification [2]. The latter estimates the failure probability based on the result of testing. However, those approaches may not reflect the past input of the software properly. This paper proposes the method to selecting test cases for the RPS software based on the profile of the state and input.

2. Reflection of internal state

Software inputs affect the program in some way, so the testing needs to reflect this effect. This section describes the way to reflect the past input sequence as the state variables.

2.1 Internal state

Past input sequences sets a system to have specific state and outputs. The state is stored in the memory of the system. The number of the states is finite, so the system can be considered as finite state machine (FSM). In this study, the variables that need to be loaded from the memory are called state variables. The state variables are the representation of past input sequences, so they need to be reflected to choosing test cases.

2.2 Test set including state variables

Test set includes inputs and state variables which represent past input sequence. Therefore, the test input does not have to include lengthy past input sequence. That is, to examine specific scenario, tester needs the information of the scenario and corresponding input sequence, but it is equivalent to test one test set and the test process can be simplified.

In addition, if several different past input sequences lead the same state, testing of each input sequence is treated as the same process. That is, the number of testing can be reduced.

3. Selecting test cases

In this section, the method to select the range of each variable in the test set is described. It will reduce the required number of test cases.

3.1 Determining the profile of Variables

The test set has several variables, constructing multi-dimensional space. Each variable has its own possible range, so the number of the required test set for exhaustive test will increase by 2^k as the number of k-bit variables increases (Fig. 1).

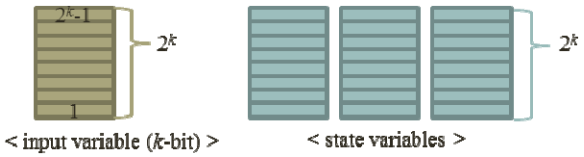

Fig. 1. Conceptual diagram of multi-dimensional input space.

However, those variables are related to each other, so the input space can be reduced. The range of each variable can be obtained by reflecting plant dynamics and the relationships of each state variable.

3.2 Determining the profile of Input

The state of the system is determined by the plant dynamics. The state variables can be said to be paired, representing a possible state of a system. Next state of an FSM is determined according to the present state and input. That is, input variables determine the next value of several state variables. However, paired state variables related to the process value might limit the range of the input by the plant dynamics. That is, if the state is determined, we can obtain possible range of the input (Fig.2.).

If we can identify the state i and p_i which is the frequency of state i, we can obtain the range of input and $q_{i,j}$, which is the conditional probability of each input values. Then the success of one test set will reveal

$p_i \times q_{i,j}$ of fault free portion. According to the previous study [2] the values of $q_{i,j}$ is identical to each other, then $q_{i,j} = 1/p_i$, because the sum of $q_{i,j}$ is one.

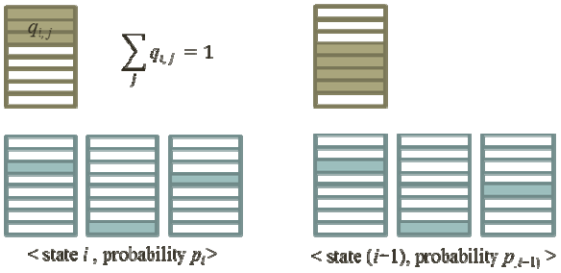

Fig. 2. Relationship of state and the range of input.

3.3 Determining the profile of state

The state of a system is represented as paired state variables, as stated. However, there are state variables related to the process value and the ones which are not related to the process value. In case of RPS software, only one pair is related to the process value.

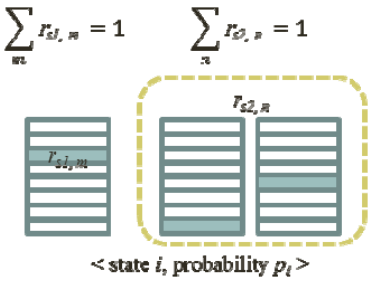

Fig. 3. State variables related to the process value (indicated as dotted line) and the ones which are not related to the process value.

The values of two kinds of variables independently change to each other. Therefore, the value of pi can be calculated as $p_i = r_{s1,m} \times r_{s2,n} \times \ldots$, where $r_{s1,m}$ is the frequency of state variable which is not related to the process value, and $r_{s2,n}$ is the frequency of set of state variables which are related to the process value.

4. Summary and conclusion

Software failure probability quantification is important factor in digital system safety assessment. In this study, the profile of paired state variables and input was obtained by reflecting plant dynamics and characteristics of digital system. The software failure probability can be estimated according to the profile of test set. Furthermore, the process of testing could be simplified and the number of test set is small enough to perform exhaustive test.

REFERENCES

[1] G.Y. Park, et al., Fault Tree Analysis of KNICS RPS Software, Nuclear Engineering and Technology, Vol. 40, No. 5, pp. 397-408, 2008.
[2] H.G Kang, et al., Input-profile-based Software Failure Probability Quantification for Safety Signal Generation Systems, Reliability Engineering and System Safety, Vol. 94, pp1542-1546