Transactions of the Korean Nuclear Society Autumn Meeting
Pyeongchang, Korea, October 30-31, 2014

Finite test sets development method for test execution of safety critical software

Sung Min Shin [a], Hee Eun Kim [a], Seung Jun Lee [b], Hyun Gook Kang [a*]
[a] *Department of Nuclear and Quantum Engineering, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Republic of Korea*
[b] *Integrated Safety Assessment Division, Korea Atomic Energy Research Institute, Daedeok-daero 989-111, Yuseong, Daejeon, 305-353, Republic of Korea*
[*]*Corresponding author: hyungook@kaist.ac.kr*

1. Introduction

These days, most of Instrumentation and Control (I&C) systems in Nuclear Power Plants (NPP) are being digitalized in response to the extended features of digital systems and difficulties of supplying analog components. Digital I&C systems can be divided into two: microprocessor based system and Programmable Logic Device (PLD) based system. The PLD based system provide more reliable performance than microprocessor based one because it can process the data in parallel and tolerance to environmental conditions. The main characteristic of this PLD system is indefinite and cyclic execution. It reads inputs, computes new states, and updates output for each scan cycle.

Korea Nuclear Instrumentation and Control System (KNICS) has recently developed a fully digitalized Reactor Protection System (RPS) based on PLD. As a digital system, this RPS is equipped with a dedicated software. The Reliability of this software is crucial to NPPs' safety where its malfunction may cause irreversible consequences [1] and affect the whole system as a Common Cause Failure (CCF). To guarantee the reliability of the whole system, the reliability of this software needs to be quantified.

There are three representative methods for software reliability quantification, namely the Verification and Validation (V&V) quality-based method, the Software Reliability Growth Model (SRGM), and the test-based method [2]. The V&V method has been utilized for this safety critical software [3-5], while SRGM has difficulties because of lack of failure occurrence data on developing phase. For the safety critical software, however, failure data cannot be gathered after installation in real plant when we consider the severe consequence. Therefore, to complement the V&V method, the test-based method need to be developed.

Some studies on test-based reliability quantification method for safety critical software have been conducted in nuclear field [6-9]. These studies provide useful guidance on generating test sets. An important concept of the guidance is that the test sets represent 'trajectories' (a series of successive values for the input variables of a program that occur during the operation of the software over time) in the space of inputs to the software [7]. Actually, the inputs to the software depends on the state of plant at that time, and these inputs form a new internal state of the software by changing values of some variables. In other words, internal state of the software at specific timing depends on the history of past inputs. Here the internal state of the software which can be changed by past inputs is named as Context of Software (CoS). In a certain CoS, a software failure occurs when a fault is triggered by some inputs. To cover the failure occurrence mechanism of a software, preceding researches insist that the inputs should be a trajectory form. However, in this approach, there are two critical problems. One is the length of the trajectory input. Input trajectory should long enough to cover failure mechanism, but the enough length is not clear. What is worse, to cover some accident scenario, one set of input should represent dozen hours of successive values. The other problem is number of tests needed. To satisfy a target reliability with reasonable confidence level, very large number of test sets are required. Development of this number of test sets is a herculean task. Therefore, another approach to cover software context without trajectory form of input is needed.

This study proposed a test set generation method for PLD based safety critical software without trajectory form of input sets. This method also considered the test coverage which was hard to deal with trajectory form based approach. To substitute trajectory form of input sets, possible ranges of variables for each situation should be identified. For this purpose, assigned range of each variable, logical relations between variables, plant dynamics under certain situation, and characteristics of obtaining information of digital device are considered. By considering above factors a CoS and an input set can be expressed as combination of single values not series of successive values of variables related. When the CoS and input set are expressed as combination of single values, testing time may take only few subsecond for one set. By adopting this method, for a simple and straightforward software like RPS, even exhausted test would be conducted.

To describe the proposed method, basic concepts of the method and dependence between variables are explained in chapter 2. In chapter 3, the feasibility of the proposed method is shown by adopting one trip logic of RPS

2. Methodology for finite test set development

2.1 Basics of proposed method

The output of software can be altered by H/W conditions or CoS, though identical inputs are applied [7]. In this study, however, the effect induced by the hardware conditions of running software such as aging of electric elements are not considered. This study is just considering logical errors inside of software. When the effect of hardware is excluded and identical input is applied to the identical CoS, same output will be reproduced. Thanks to this deterministic characteristic of software there is no need to conduct repeated test for one test set.

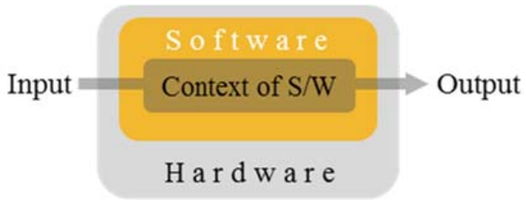

Figure 1 Components of software test

The fundamental concepts of proposed method are like below. Practically, a CoS consist of certain values of corresponding variables. Here the variables composing CoS, generated and saved ones at the last scan time inside of the software, are named as Context Variables (CVs). If the several different past input sequences lead the same context, the last CoSs are regarded as identical ones. In this point of view, the past input sequence has no meaning. To find out all possible CoSs in systematic way, what really mattered is the possible range of each variable mutually. When the possible range of each CV is identified, all realizable contexts can be expressed by combinations of values of each CV.

The variables composing input sets, newly acquired ones for current scan time from outside of software, are named as Input Variables (IVs). Based on a certain CoS, each IV also has its limited range. Actually, some IVs represent values of process parameters which are converted through Analog to Digital Converter (ADC). Because of continuity of process parameter and the converting characteristic of ADC such as scan interval, the possible difference of sequent value of an IV is limited [10]. Except those variables representing process parameters, the others are just Boolean type of variables in general. Their possible states are true or false, so it doesn't matter to set input sets. When the possible ranges of each IV is identified, similar to CVs, input sets can also be expressed by combinations of IVs

2.2 Variables for CoS and input set

As mentioned above, a CoS and the input set to this CoS can be expressed as the form of combination of values related. For this approach, the variables for CoS and input set should be classified first. And then all possible CoSs, without omission, should be investigated. Thereafter, based on each CoS, input sets can be developed.

The variables for CoS and input set can be discriminated according to the definition of CV and IV described in 2.1. The variables acquired from outside of software for this scan time are IVs, and the variables saved inside of software at the last scan time are CVs.

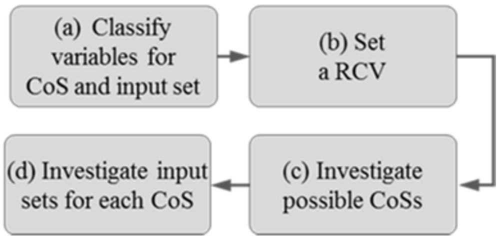
(a) Classify variables for CoS and input set
(b) Set a RCV
(c) Investigate possible CoSs
(d) Investigate input sets for each CoS

Figure 2 Process of test set development

To investigate all possible CoSs, a concept of Reference Context Variable (RCV) is adopted. RCV is like a flag to restrict the possible ranges of other CVs. For a better understanding, a variable representing process parameter of previous scan time, such as pressure or temperature, can be an example of this RCV. In normal operation, trip setpoint which is one of the CVs cannot cross the previous pressure or temperature [11]. To take certain value of RCV, its original range and resolution should be identified. The assigned range of a RCV can be deduced from maximum and minimum value of it, and the resolution will be the assigned range divided by assigned memory for this variable.

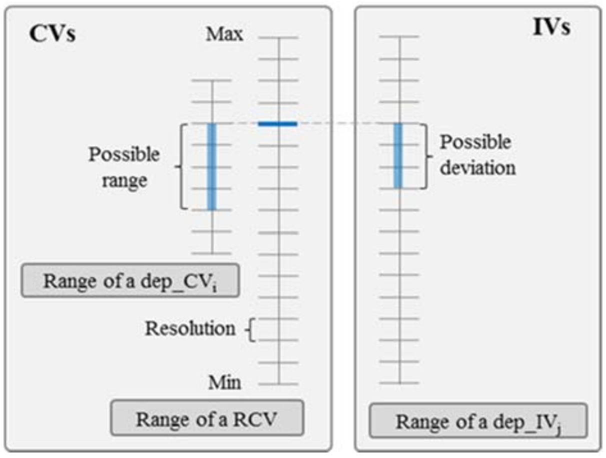

Figure 3 Dependency between RCV and other variables

When a value of RCV is set, the possible ranges of other CVs are limited because of their dependencies to this RCV as described before. Basically, here the dependency is logical relation between variables programed in the software. If some CVs have no relation with RCV, all assigned range need to be considered. After choosing the RCV, by changing the value of this RCV from minimum to maximum over the assigned range, possible ranges of other dependent CVs associated with a certain RCV value can be identified. Consequently, mutually possible range between CVs can be checked, and by using the values in the ranges identified, the possible CoSs can be formed.

In regards to IVs, except some Boolean type of variables such as permission and reset, remaining IV is also representing process parameter because PLD is the system which monitors and compares certain parameter continuously under certain conditions. Based on the RCV, the value of this IV cannot go further beyond the certain deviation during scan interval because of its physical continuity. The possible deviation can be determined in consideration of plant dynamics and characteristics of ADC [10]. Plant dynamics is associated with slope of transition, and the characteristic of ADC is associated with scan interval. The consecutive value can go further from the previous value, when the process parameter changes drastically and it is scanned sparsely as well. To get the possible drastic slope, simulation code for accident scenario can be utilized.

For the test, when one set of CoS is formed, based on this CoS, all logically possible input sets should be applied, respectively. And then to one another CoS, all the other possible input sets to this new CoS should be applied again. The important thing here is that test input does not have to include lengthy past input sequence but can be expressed as a combination of variables related, thus the test process can be simplified.

2.3 Test set development

When the dependencies between variables are considered, the numbers of test sets for a certain value k of RCV and total test sets can be expressed like equation (1) and (2), respectively.

$$N_k = \left[\left\{\prod_{i=1}^{n} N(CV_{in_i}) \times \prod_{i=1}^{m} N\left(CV_{dep_i} \middle| RCV_k\right)\right\} \times \left\{\prod_{i=1}^{l} N(IV_{in_i}) \times \prod_{i=1}^{k} N(IV_{dep_i}|RCV_k)\right\}\right] \quad (1)$$

$$N_{test} = \sum_{k=min}^{max} N_k \quad (2)$$

Where,
$N(variable_i|RCV_k)$: Number of possible states of variable i under certain value k of the RCV
CV_{in_i} : Independent context variable
CV_{dep_i} : Dependent context variable
IV_{in_i} : Independent input variable
IV_{dep_i}: Dependent input variable
N_{test}: Total number of tests

When the dependencies between variables are considered, the number of total test sets can be reduced a lot compare with the case which is not consider the dependency because, in general, the possible range of each CV under certain situation may be restricted to some portion of assigned range. However, the total number of test sets might still too big to test. Main contributor for this big number of test sets might be the resolution of RCV. Basically, to control the system precisely, resolution of variables are very fine. On account of this fine resolution, the total number of test sets may be thousands times more than the coarse one. If the total number of test sets are too big, total testing time may take few months or more even though one test takes few subsecond. This is not practical. In such cases, there is no advantage comparing with the trajectory form of input based approach. There are two means to reduce the number of test sets in reasonable way because this approach considers logical relation between variables.

First mean for test sets reduction is application of safety point of view. In the safety point of view, failure is incorrect output when a demand comes. So, some situations which not require safety action such as trip signal generation are out of interest. In this research, it is distinguishable whether some situation demand safety action or not because the logical relation between variables and the range of each variable are being considered. When only the cases requiring safety action are extracted, total number of test sets can be reduced.

The second mean is equivalence partitioning. Equivalence partitioning is the process placing possible test sets into classes [12]. Usually it is the input that is partitioned. However, according to the software, output also can be partitioned. In this study, to cover all CoS, input sets can be partitioned. However, if necessary, CoS also can be partitioned. The criteria for partitions are usually derived from expected result of the input specified in requirement. Each partitioned shall contain a range of values, chosen values in a partition can reasonably be expected to be treated by the software in the same way [a]. Therefore, when one representative value is applied, all values in this partition are considered as covered. In this manner, total number of test sets can be reduced but exhausted test cannot be accomplished.

3. Application to the RPS software

3.1 Variables in objective trip logic

The proposed method was applied to the RPS software to show its feasibility. For this purpose, most complicated trip logic was selected and the variables of it were investigated. In RPS, there are 19 trip logics. All the logics can be divided into 3 categories according to their setpoint type [1]: fixed trip setpoint, variable trip setpoint by manual reset, and variable trip setpoint by automatic rate-limiting. Variable type logics are more complicate than fixed one because, along with the name, its setpoint might be changed. Among variable type of trip logics, only pressurizer pressure low trip (PZR PR Lo Trip) logic has operator bypass function additionally. Thus, this trip logic is selected as the objective for application because it is considered as most complicated one.

Fig. 4 shows the set point variation logic for PZR PR Lo Trip [11]. Basically, this logic will generate a trip or pretrip signal when the system pressure decreases and reaches to the trip or pretrip setpoint. However, these setpoints can be changed depending on the system pressure and operator reset as shown in Fig. 4.

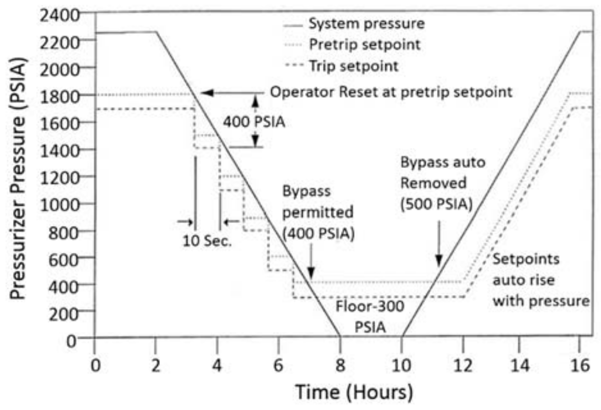

Figure 4 Trip logic for pressurizer low pressure trip

When system pressure rises away from setpoint, setpoint will chase the system pressure after 400 psi. But when the system pressure falls toward the setpoint, the setpoint will not be changed. To drop the setpoint, operator should generate reset signal. When the reset signal is generated, the trip setpoint will be drop 400 psi from the system pressure at that timing and cannot be changed again for 10 sec. If the system pressure drops under 400 psi, bypass is permitted. If the operator make bypass signal when bypass is permitted, trip logic will be bypassed and cannot make trip signal. And when system pressure exceed 500 psi, bypass permission will be removed automatically.

Among the variables inside of PZR PR Lo Trip logic, when the variables for mode selection such as test and real mode, the fixed constants, and the intermediate ones which are automatically calculated according to input values are excluded, remaining variables can be summarized like table 1

Table 1 Summarized variables in PZR PR Lo trip logic

Context variable		Input variable	
name	Type	name	Type
Previous pressure	WORD	Current pressure	WORD
Trip setpoint	WORD	Bypass from MCR	BOOL
Rest delay time	WORD	Bypass from RSR	BOOL
		Reset from MCR	BOOL
		Reset from RSR	BOOL
		Module error	BOOL
		Channel error	BOOL

3.2 Dependency in objective trip logic

Among CVs, the value of previous pressure can be a RCV. Minimum, maximum value, assigned memory, and dependencies of other variables to this RCV are shown in table 2. This value is obtained by dividing the difference between maximum and minimum value by assigned memory.

Table 2 Dependency of each variable to previous pressure

Base Context variable	Previous pressure		
	Min (psi)	Max (psi)	Memory (Word)
	100	2,940	26,400
Dependent variable	Trip setpoint (CV)		
	Current pressure (IV)		
Independent variable	Reset delay time (CV)		
	Module error (CV)		
	Channel error (CV)		
	Bypass from MCR (IV)		
	Bypass from RSR (IV)		
	Reset from MCR (IV)		
	Reset from RSR (IV)		

s

Dependent variables to the previous pressure (RCV) have relations like below. Basically, trip setpoint cannot exceed previous pressure and cannot be less than 400 psi from the previous pressure. And the programed minimum and maximum value of trip setpoint are 300 and 1780 psi, so it will be readjusted as these programed values even if they are calculated differently based on system pressure. As mentioned above, dependent input variable (current pressure) cannot exceed certain deviation. Kang [10] investigated the possible deviation of system pressure for sequent scan timing in consideration of plant dynamics under several hole sizes of Loss of Coolant Accident (LOCA) scenario and ADC characteristics about scan time and memory. Currently, processing time of BP should be shorter than 50 ms, so 50 ms can be considered as the scan interval. When we assume that the biggest hole size (0.7610 m) of LOCA scenario is the accident which can make most drastic pressure change and the memory of ADC is 12 bit, according to the reference, the current pressure can be dropped about 3.3 psi from the previous pressure. For the possible deviation of current pressure, this result can be simply quoted to calculate the number of finite test sets. All the other independent variables, except reset delay time, are Boolean type. These values have just two states. The resolution of reset delay time will be 50 ms corresponding to the scan interval.

3.3 Finite test sets

The total numbers of test sets are calculated and shown in figure 5 when just dependency, safety point of view, IV partitioning, and IV & CV partitioning are considered, respectively.

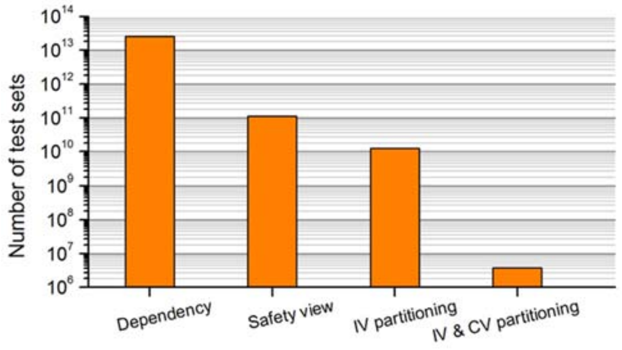

Figure 5 Total number of test sets according to different approach

For the case which just considers dependency between variables, the previous pressure (RCV) was changed with 0.1 psi interval (resolution) from 2940.0 psi to 300 psi, and all possible states of other dependent variables to each value of previous pressure are investigated. In this case, total number of test sets is 2.59E13. This number is too large to test all. Main contributors of this large number are 4000 states of trip setpoint for each previous pressure (1780.0~700.0 psi) and 200 states of rest delay time for all previous pressure. Here the variable for reset delay time is only for three kinds of trip logics: pressurizer low pressure trip, steam generator low water level trip, and steam generator low coolant flow trip. Among them, pressurizer low pressure trip has the longest delay time.

In the safety point of view, the number of sets which requires trip signal generation is extracted. There are three kinds of cases requiring trip signal. Two are module error and channel error. They are just boolean type. The other case is crossing trip setpoint of current pressure. The all sets that when the current pressure drops 3.3 psi from the previous pressure, it cross the trip setpoint are counted. The total number of test sets for this case is 1.13E11. Practically, this number is still large to test all.

Equivalence partitioning for IV domain was considered to reduce the number of test sets. 33 or less than 33 states for current pressure for each current pressure are reduced to 2. Actually, except the variable for current pressure, all other variables are boolean type. So, partitioning of IV domain is not very effective for this example logic. In this case, total number of test sets is 1.26E10.

If CV domain is partitioned additionally, the total number of test sets decreases drastically up to 3.80E6. The resolution for previous pressure (RCV) was maintained, but all possible states for trip setpoint and reset delay time do not need to be applied. When a trip signal is generated well for one set which is consisted of representative value of trip setpoint and reset delay time from each partition, this result is considered as the result covering the all values in each chosen partition. Therefore, 4000 or less than 4000 states for trip setpoint and 200 states for reset delay time are reduced to 2, respectively. When we assume that the computing time for one test set is few millisecond, it will take just few hours to calculate all number of test sets for this logic. Especially, in RPS, the fact that this trip logic is the most complicated one should be considered with the result

4. Conclusions

Trajectory form of input based approach for a safety critical software reliability quantification has limitations caused by unclear length and the numerous number of input sets required.

To address these limitations, another method to generate test sets which is consist of CoSs and input sets was proposed. If the possible ranges of each variable might be identified, each of CoS and input set could be expressed as combination form of single values of each variable not trajectory form. The possible range of each variable can be identified through logical relations between variables, plant dynamics, and characteristics of ADC. When the proposed method is applied, the test process might be simplified and shorten compare to the case of trajectory form of input.

The feasibility of this method was shown through an example trip logic in RPS. In this example, when just dependency between variables were considered, the total number of test sets was impracticable. However, by utilizing the means described here to reduce the number of test sets the reasonable number of test set could be obtained. To get the proper number of test sets, different control per variable also possible because this method considers logical relations between variables and possible resolution & ranges of them.

In case of software for straightforward and simple PLD based system, such as RPS, the proposed method expected to work properly to generate finite tests without trajectory form of input.

Nomenclature

I&C (Instrumentation and Control)
NPP (Nuclear Power Plant)
KNICS (Korea Nuclear Instrumentation and Control System)
RPS (Reactor Protection System)
CCF (Common Cause Failure)
V&V (Verification and Validation)
SRGM (Software Reliability Growth Model)
CoS (Context of Software)
CV (Context Variable)
IV (Input Variable)
ADC (Analog to Digital Converter)

RCV (Reference Context Variable)
LOCA (Loss of Coolant Accident)

REFERENCES

[1] G.Y. Park, K.Y Koh, E.Y. Jee, P.H. Seong, K.C. Kwon, D.H. Lee "Fault Tree Analysis of KNICS RPS Software" Nuclear Engineering and Technology, Vol. 40 No.5, pp. 397-408, 2008.
[2] T.L. Chu, M. Yue, G. Martinez-Guridi, J. Lehner "REVIEW OF QUANTITATIVE SOFTWARE RELIABILITY METHODS" Brookhaven national Lab, 2010.
[3] N. Fenton, M. Neil, W. Marsh, P. Hearty, D. Marquez, P. Krause, M. Rajat "Predicting software defects in varying development lifecycles using Bayesian nets" Information and Software Technology, Vol. 49, No.1, pp. 32-43, 2007
[4] N. Fenton, M. Neil, D. Marquez "Using Bayesian networks to predict software defects and reliability" Journal of Risk and Reliability, Vol. 222, No. 4. pp. 701-712, 2008.
[5] H. Eom, G. Park, S. Jang, H. S. Son, H. G. Kang "V&V-based remaining fault estimation model for safety–critical software of a nuclear power plant" Annals of Nuclear Energy, Vol. 51, pp. 38–49, 2013.
[6] J. May, G. Hughes, A. Lunn. "Reliability estimation from appropriate testing of plant protection software" Software Engineering Journal, Vol.10, No. 6, pp. 206-218, 1995.
[7] T.L. Chu, M. Yue, G. Martinez-Guridi, J. Lehner "Development of quantitative Software Reliability Models for Digital Protection Systems of Nuclear Power Plants. U.S. NRC, 2011.
[8] S. Kuball, JHR. May, "A discussion of statistical testing on a safety-related application". Journal of Risk and Reliability, Vol. 221, No. 2, pp. 121-132, 2007.
[9] B. Littlewood, L. Strigini, "Guidelines for Statistical Testing" Centre for Software Reliability at City University, 1997.
[10] 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 & System Safety, Vol. 94, No. 10, pp. 1542-1546. 2009.
[11] J.G. Choi, D.Y. Lee "Development of RPS Trip Logic Based on PLD Technology. Nuclear Engineering and Technology, Vol. 44, No. 6, pp. 679-708, 2012.
[12] I. Burnstein "Practical Software Tesing" Springer, 2002.