TEST BASED RELIABILITY ASSESSMENT METHOD FOR A SAFETY CRITICAL SOFTWARE IN REACTOR PROTECTION SYSTEM

Sung Min Shin*, Jaehyun Cho and Wondea Jung
Korea Atomic Energy Research Institute
989-111 Daedeok-daero, Yuseong-gu, Daejeon, 305-353, Republic of Korea
smshin@kaeri.re.kr; chojh@kaeri.re.kr; wdjung@kaeri.re.kr

Seung Jun Lee
Ulsan National Institute of Science and Technology
UNIST-gil 50, Ulsan 689-798, Republic of Korea
sjlee420@unist.ac.kr

ABSTRACT

Digitalization of the instrumentation and control systems in nuclear power plant entails some new features which do not exist in analog systems. When the new features are applied in safety critical systems, the risk stemming from them should be quantified properly to assure the reliability of the entire power plant. Among the new features, software is considered as the most important factor because it can cause common cause failure to many redundant systems. Therefore, in this work, a test based reliability assessment method for a safety-critical software is developed.

An output of a software is determined by not only inputs but also the internal state of the software at the time. In the case of safety critical software developed for usage in nuclear power plant, it is possible to directly investigate probable internal states thanks to detailed design specification and programing features, and the probable inputs also can be identified based on the physical linearity of each process parameter and hardware characteristics related to data acquisition. In this work, a development method for practical exhaustive test case consisting of investigation of internal state and probable input sets to a specific internal state are suggested. In addition to this, software logic simulator for execution of software logic test and the reliability quantification method based on the test result are developed. The feasibility of the suggested method is demonstrated via a case study.

Key Words: Safety-critical software, Digital I&C, Reactor protection system

1 INTRODUCTION

Currently, most of worldwide I&C systems in nuclear power plants (NPPs) are being digitalized due to obsolescence of safety-grade analog components. This shift entails adoption of new features which did not exist in analog systems. Although the features are expected to contribute to the enhancement of both efficiency and economy, from the safety point of view, the risk caused by the new features should be analyzed in an appropriate framework to ensure the dependability of the entire NPP.

Among the new features, software is considered as the most important feature due to the possibility of its common cause failure (CCF). In the case of identical software installed in multiple redundant systems, a failure of the software can cause loss of the entire system [1]. In this circumstance, the reliability of the software must be properly quantified to guarantee the reliability of the NPP.

There have been many approaches made to quantify the reliability of software. Software reliability growth model (SRGM) is the most common method for this purpose. The SGRM basically estimates the remaining defects based on debugging history. However, it is difficult to apply this method to the safety-critical software because not only sufficient failures sets are unavailable but consequence of system failure is also unacceptable [2]. Bayesian belief network (BBN) is another promising approach assessing the quality of software lifecycle activities from requirement to validation phase. Although this method is currently being applied to the safety-critical software, the obtained result by this method may have large uncertainty. Thus, the BBN result need to be complemented or verified by other methods. Recently, to demonstrate the integrity of safety-critical software, black-box test which feeds inputs and examines outputs whether they succeeded or failed is considered based on trajectory-form-of-input (a series of successive values for each input variable). An output of a software is determined by not only inputs but also an internal state of the software at the time. This method lets the internal state be changed by the applied trajectory-form-of-inputs which are randomly sampled from operational profile [3-4]. In this approach, however, there are some difficulties to develop enough number of test cases to demonstrate required reliability of safety-critical software with reasonable confidence level.

In this study, a novel test based reliability assessment method for a safety-critical software is proposed. This method facilitates obtaining the practical exhaustive test cases to demonstrate the software integrity from the safety point of view. The exhaustive test cases are the combination of possible internal states and possible inputs to each internal state. The possible range of each internal state and input set can be identified based on software design specification, programming features, physical linearity of each process parameter, and hardware characteristics related to data acquisition. To show the feasibility of this approach, actual test cases are developed for an example RPS source code and executed with logic simulator developed.

2 FEATURES OF SAFETY-CRITICAL SOFTWARE AND INVESTIGATION ON CORE VARIABLES

2.1 Features of safety-critical software

To quantify the reliability of the safety-critical software based on test, test related variables should be identified first. The most important priority of safety-critical software is functional integrity. Therefore, all the software design specifications and programing features focus on realization of it, and such information can be utilized to identify core variables for test execution.

One of the features of safety-critical software is sequential performance to finish the execution of duty within determined time. To satisfy this requirement, safety-critical software does not allow interrupt or loop logic and keeps one-way logic flow. The sequential performance is programmed by graphical language such as function block diagram (FBD) or ladder diagram (LD), and these languages are very efficient to investigate variables which affect output value.

One another feature of safety-critical software is repetition of the sequential performance with fixed time interval. It reads input, computes new state, updates, and reads input again for the next scan cycle. The important thing of this feature is that the software determines output at the end of each scan cycle using the inputs scanned for that. As this feature, test can be executed using just one value for each variable, with no need for trajectory-form-of-input.

2.2 Investigation and classification of variables

To identify the core variables, definition and classification of each variable are essential. Fig. 1 shows a conceptual configuration of variables in safety-critical I&C software used in a reactor protection system.

The concerning output variable related to software integrity is naturally decided when a target safety function is determined. From the safety point of view, only variables which affect this output need to be considered, and such variables can be found through backward tracing. In Fig. 1, output C1 depends on input C1, C2, and C3. Input C1 and C2 are connected to output A2 and B2, respectively. Thus, during the development of test cases, actual values for output A2 and B2 do not need to be considered, but the variables that affect output A2 and B2 should be identified. Among the variables that affect output A2, input A1~A3 are scanned from the outside of the boundary of application software, but input A4~A5 are the stored values. Regarding the output B2 in function block B, input B1~B2 are transferred from function block A, but input B3 is also called from saved memory. According to the above investigation, variables in a safety-critical software can be divided into following three categories:

1. Input variable (IV): value of it is obtained for current cycle from the outside of software
2. Internal-state variable (SV): value of it is saved at inside memory arranged at the last scan cycle
3. Intermediate variable (MV): value of it is generated during logic execution to give input to other function block

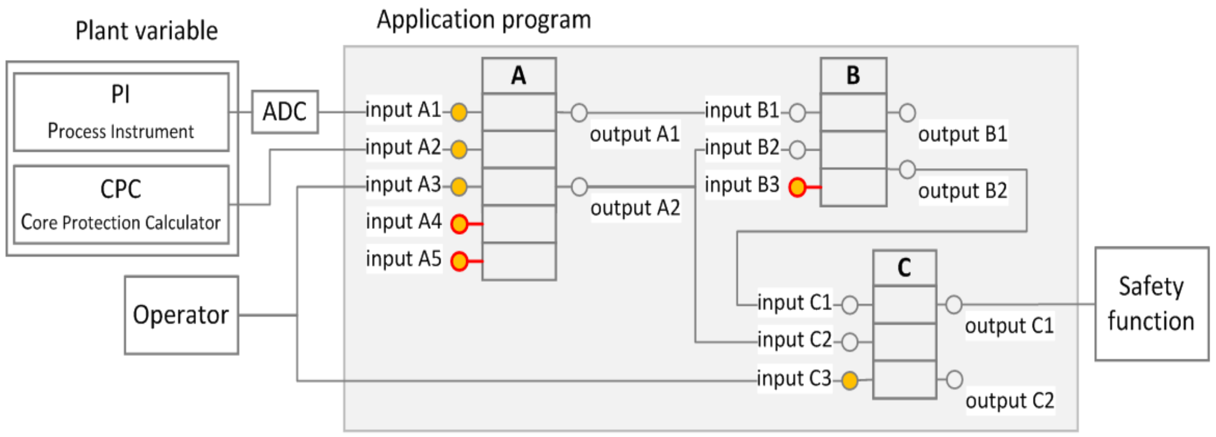


Figure 1. Configuration of variables in safety-critical I&C software

3 DEVELOPMENT OF EXHAUSTIVE TEST CASE

3.1 Investigation on internal state

The probable internal state of software is combination of values for each SV. The possible values for each SV is decided by the assigned range, which can be determined by minimum and maximum value, and resolution. Such information is given in the software design specification. In case all the SVs are independent of each other, the combinations of all the values in the assigned range need to be considered. In the real world, however, there are some dependencies between SVs. Thus, when a value for a specific SV is set, the possible range of another SV can be limited. In this manner, the probable internal state of a software can be investigated by identifying assigned range and resolution and by considering dependencies between SVs.

3.2 Investigation of input set

For the test execution, after confirmation of probable internal states, possible input sets to a specific internal state should be investigated. The input sets are basically composed by IVs. The input variables can be divided into two, which are plant variable and operator input as shown in Fig. 1. The values of plant variable are generated by process process instrument (PI) or core protection calculator (CPC) according to the instrumentation.

Although the value of operator input can be determined according to plant situation considered, the values of plant variables that are supposed to be obtained for the current scan cycle has certain possible deviations from the previous value because of physical linearity. This relation is well described in Kang's study [5]. Under a specific resolution of each variable, the possible deviation of current value for each plant variable depends on the scan interval, scan timing, and plant dynamics. Fig. 2 shows this relation. When a plant variable changes drastically and it is scanned sparsely as well, the consecutively obtained value can go further from the previous point. By considering this relation, the test cases that will not occur

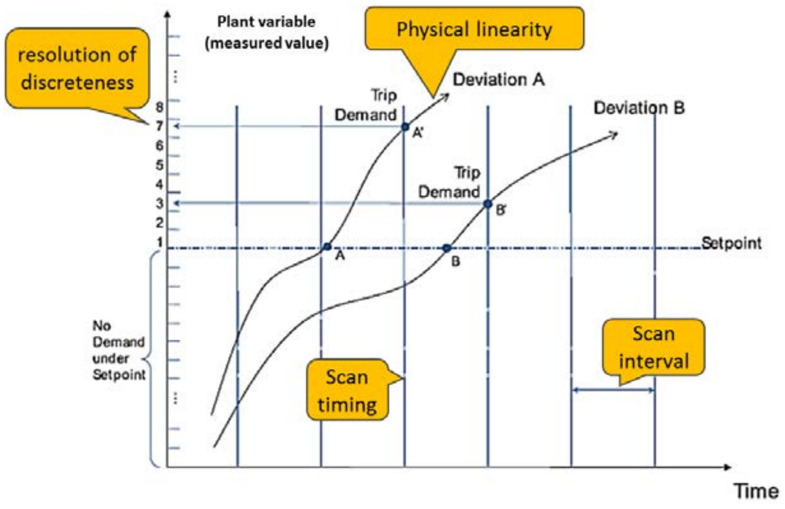


Figure 2. Scan timing and demand generation [5]

in actual use can be excluded, and all probable cases can be tested.

3.3 Exhaustive test cases

An output of software has deterministic characteristics. When the same input set is applied to the same internal state, same output will be generated, no matter how many times the test case is executed. Therefore, to confirm the software integrity for a specific test case, just single execution is required. Based on this premise, the number of exhaustive test cases can be calculated from the formula below:

$$N_{total} = \sum_{i=1}^{m} N(input\ set|internal\ state_i) \tag{1}$$

N_{total}: Total number of exhaustive test cases

m: Number of possible internal states

$N(input\ set|internal\ state_i)$: Number of possible input sets to a specific internal state i

4 SOFTWARE RELIABILITY QUANTIFICATION

When an acceptable number of exhaustive test cases is given all the test cases can easily be executed and, according to the test results, the completeness of the software also able to be confirmed. However, in case of tremendous number of exhaustive test cases is given, an efficient approach should be taken to prove required the level of software reliability.

A kind of test coverage, defined like equation (2), can be utilized as an index for test progress. This index indicates number of test cases executed against total number of exhaustive test cases.

$$\mathrm{C} = \frac{number\ of\ test\ cases\ executed}{total\ number\ of\ exhaustive\ test\ cases} \tag{2}$$

When each test case is considered as a sample space, the entire failure probability of target software can be expressed like equation (3).

$$\theta_t = \sum \theta_{j|i}(q_{j|i} \times p_i) \tag{3}$$

p_i: Existence probability of internal state *i*

$q_{j|i}$: Conditional existence probability of input set *j* given internal state *i*

$q_{j|i} \times p_i$: Existence probability of test case expressed by internal state *i* and input set *j*

$\theta_{j|i}$: Failure probability of the test case expressed by internal state *i* and input set *j*

When no fault is detected after a test for internal state *i* and input set *j*, it can be said that the corresponding portion to $q_{j|i} \times p_i$ is error-free, and validation of required level of software reliability can be achieved by testing the software in the order of probable test cases. Fig. 3 shows the general relation between test coverage and confirmed software reliability when the software test is executed through the suggested approach under an assumption that no failure is detected.

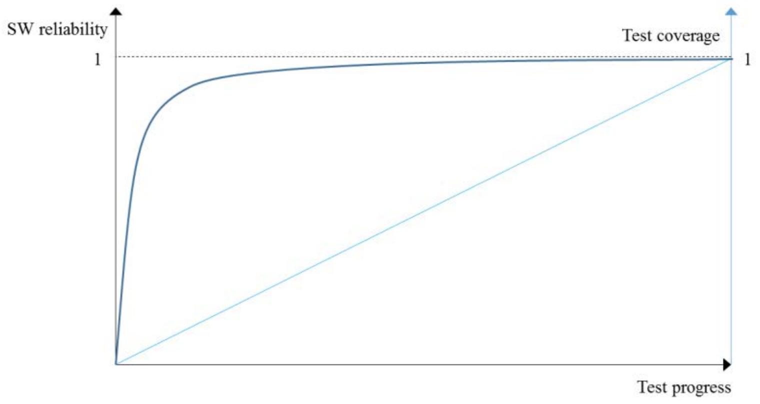


Figure 3. Confirmed software reliability according to the test coverage

For this approach, the existence probability for each test case needs to be quantified. Regarding the existence probability for each internal state, for the values of independent SVs, uniform distribution of probability can be considered in the assigned range. Whereas probable values for each dependent SV are limited by each other as mentioned before. The existence probability of particular combination between dependent SVs can be identified by getting conditional probability of respective value of each dependent SV in sequence, after determining an order between dependent SVs.

In the investigation on existence probability for each input set, the result of thermal-hydraulic code analysis can be utilized with the considerations from the safety point view. What we want to prove through the test is whether the software can generate safety signal or not when a plant variable meets the conditions for a safety function. Therefore, concerning region related to the plant variable is just some beyond trip set point (TSP). When all PSA initiating events and its scenarios are considered, the existence probability of each discreteness value for plant variables beyond the TSP can be identified.

5 CASE STUDY

The Korea Nuclear Instrumentation and Control System (KNICS) has developed a fully digitalized RPS based on a programmable logic controller (PLC) [6]. As a digital system, this RPS is also equipped with dedicated software. The proposed method was applied to an example software logic, pressurizer low pressure trip (PZR low_P trip) logic, to demonstrate the feasibility of the proposed method. The detailed description about this trip logic can be found in Choi's study [7]. As a simple case study, exhaustive test cases only for the full power operation condition for level 1 PSA are developed and tested using the software logic simulator developed.

The trip function will be activated by trip signal generated, so a variable indicating trip signal is determined as the concerning output. Through analysis of FBD source code of this software, the IVs and SVs related to the trip signal are identified and summarized in Table I.

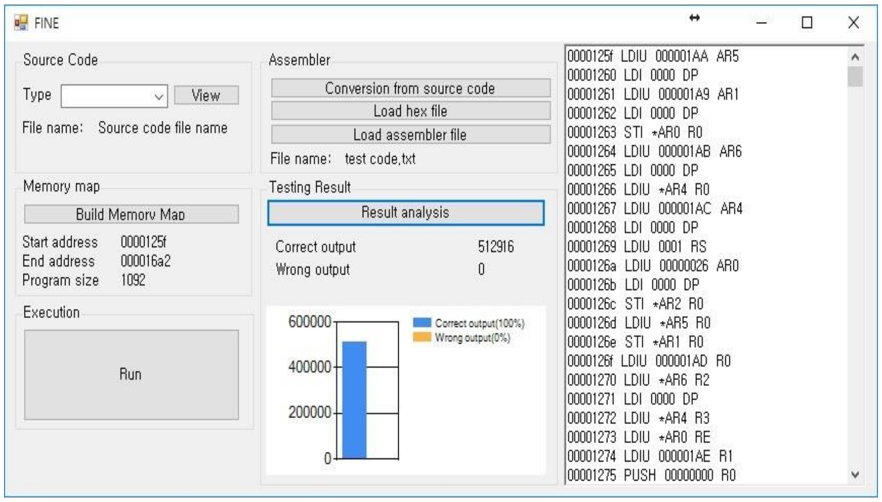
Table I. Core variables in PZR low_P trip

Variable category	Name	Description
State variable	TSP	Trip set point for trip signal generation
	Bypass permission	Bypass is permitted according to pressure range
	Delay time	Elapsed time since TSP reset
Input variable	Mode 1	Operation mode selection
	Mode 2	
	Mode 3	
	Pressure	Plant variable
	Hardware error 1	Error detection
	Hardware error 2	
	Signal error	
	Bypass	Trip bypass by operator
	Reset TSP	TSP reset by operator

In case of full power operation conditions for level 1 PSA, bypass of trip and reset of TSP are not activated, so the values of internal variables related to these features are not changed. Therefore, under this conditions, internal state need to be addressed is only one case that the combination of default values for each SVs. Actually, during full power operation conditions, all the internal state for each trip logic are similar to this case.

To develop the input sets to this internal state, probable combinations of each input variable should be identified. Among the IVs, probable values for all the IVs except pressure can be identified based on the conditions of full power operation, whereas thermal hydraulic code analysis needs be conducted to identify probable values for pressure. If we make input profile for all PSA initiating events and its scenarios, the possible deviation of pressure value from the TSP can be found. Regarding a target PSA model (OPR 1000), 18 initiating events and a number of scenarios in each initiating event are investigated. Based on the hardware information in software design specification (30,000 resolution of discreteness and 50 ms of scan interval), it is confirmed that maximum 2 discreteness value can exceed from the TSP.

When the situations for trip signal generation caused by not only pressure drop but also errors are considered, total 512,820 cases are identified as the exhaustive test cases. Then all the test cases are executed by using the software logic simulator, and no failure is detected. Fig. 4 shows the test result of the logic simulator and Fig. 5 indicates the SW reliability according to the test coverage.



Type:

File name: Source code file name

Start address 0000125f
End address 000016a2
Program size 1092

File name: test code,txt

Correct output 512916
Wrong output 0

```
0000125f LDIU 000001AA AR5
00001260 LDI 0000 DP
00001261 LDIU 000001A9 AR1
00001262 LDI 0000 DP
00001263 STI *AR0 R0
00001264 LDIU 000001AB AR6
00001265 LDI 0000 DP
00001266 LDIU *AR4 R0
00001267 LDIU 000001AC AR4
00001268 LDI 0000 DP
00001269 LDIU 0001 RS
0000126a LDIU 00000026 AR0
0000126b LDI 0000 DP
0000126c STI *AR2 R0
0000126d LDIU *AR5 R0
0000126e STI *AR1 R0
0000126f LDIU 000001AD R0
00001270 LDIU *AR6 R2
00001271 LDI 0000 DP
00001272 LDIU *AR4 R3
00001273 LDIU *AR0 RE
00001274 LDIU 000001AE R1
00001275 PUSH 00000000 R0
```

Figure 4 Test result of exhaustive test case for PZR low_P trip

6 CONCLUDING REMARKS

In this work, a development method of exhaustive test case for reliability quantification of a safety-

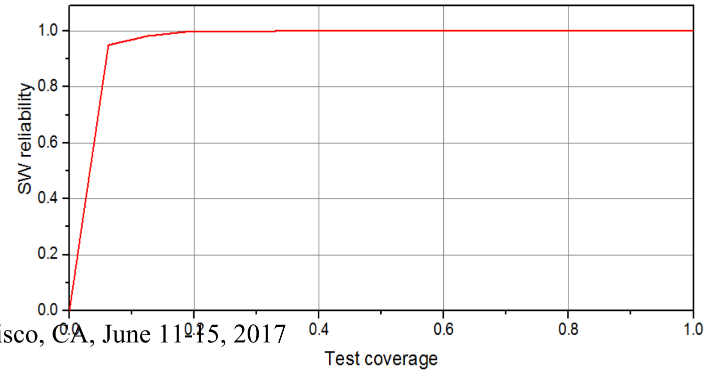


Figure 5. Reliability of PZR low_P trip logic according to the test coverage

critical software in a nuclear power plant is developed. The suggested test based software reliability quantification method can significantly remove the uncertainties present in the quantification process and can pinpoint the erroneous part through the straightforward test process, and it is even possible to prove completeness of the target software. The method is based on an assumption that the target software can be separated from others, so the independency between target software and others should be confirmed.

The quantified value of software reliability itself shows the level of software integrity and it can be applied to the reliability assessment of the entire digitalized NPP. This method can be utilized for improvement or supplementation of the software function when it is applied to the design phase and also can be utilized as an acceptance criterion at the licensing phase.

7 ACKNOWLEDGMENTS

This work was supported by Nuclear Research & Development Program of the National Research Foundation of Korea (NRF) grant, funded by the Korean Government, Ministry of Science, ICT & Future Planning (Grant Code: 2012M2A8A4025991).

8 REFERENCES

1. Kang HG, Sung T., "An analysis of safety-critical digital systems for risk-informed design," *Reliability Engineering and System Safety*, **Vol. 78**, pp. 307–14 (2002)
2. Kim M, Jang S, Ha J., "Possibilities and limitations of applying software reliability growth models to safety-critical software," Nuclear Engineering and Technology, **Vol. 39**, pp. 129–32 (2007)
3. Kuball S, May JHR., "A discussion of statistical testing on a safety-related application," *Proceeding of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, **Vol. 221**, pp. 121–32 (2007)
4. Chu T, Yue M., Martinez-Guridi G, Lehner J., "Development of Quantitative Software Reliability Models for Digital Protection Systems of Nuclear Power Plants," NUREG/CR-7044, NRC (2013)
5. Kang HG, Lim HG, Lee HJ, Kim MC, Jang SC., "Input-profile-based software failure probability quantification for safety signal generation systems," *Reliability Engineering and System Safety*, **Vol. 94**, pp. 1542–46 (2009)
6. Kwon K, Lee M., "Technical review on the localized digital instrumentation and control systems." *Nuclear Engineering and Technology*, **Vol. 41**, 447–54 (2009
7. Choi JG, Lee DY., "Development of RPS Trip Logic Based on PLD Technology," *Nuclear Engineering and Technology.*, **Vol. 44**, pp. 697–708 (2012)