# Development of Software Test-based Reliability Assessment Method for Nuclear Power Plant Safety-critical Software

**Sang Hun Lee[a], Seung Jun Lee[b], Jinkyun Park[c], Eun-chan Lee[d], and Hyun Gook Kang[a*]**

[a] Department of Mechanical Aerospace and Nuclear Engineering, Rensselaer Polytechnic Institute (RPI), 110 8th street, Troy, NY, USA, 12180

[b] School of Mechanical, Aerospace and Nuclear Engineering, Ulsan National Institute of Science and Technology (UNIST), 50 UNIST-gil, Ulsan, Republic of Korea, 44919

[c] Integrated Safety Assessment Division, Korea Atomic Energy Research Institute (KAERI), 111 Daedeok-daero, 989beon-gil, Yuseong-gu, Daejeon, Republic of Korea, 34057

[d] Korea Hydro & Nuclear Power Co., Ltd., 1655 Bulguk-ro, Gyeongju-si, Gyeongsangbuk-do, Republic of Korea, 38120

**Abstract:** Software has been used to digitalize many instrumentation and control (I&C) systems in nuclear power plants (NPPs). Since software failure induces the common cause failure of the processor modules, the reliability of the software used in NPP safety-critical I&C systems must be quantified and verified with proper test cases and environment. In this study, a software testing method using the simulation-based software test-bed is proposed. In the test-bed, the microprocessor architecture of the programmable logic controller (PLC) used in NPP safety-critical applications is emulated and the execution behavior of the microprocessor at each machine instruction line is captured. The effectiveness of the proposed method is demonstrated with the safety-critical trip logic software of a fully digitalized reactor protection system (IDiPS-RPS). The software test cases are developed in consideration of the digital characteristics of the target system as well as the plant dynamics to represent the possible states of software input and internal variables that contributes to generating its dedicated safety signal. The method provides a practical way to conduct software testing in order to prove the software to be error-free while effectively reducing the software testing effort by emulating the PLC behavior in machine-level compared to existing software testing methods.

**Keywords:** Nuclear Power Plant, Digital I&C System, Safety-critical Software, Software Testing.

## 1. INTRODUCTION

Nuclear power plants (NPPs) employ several safety systems to protect the public from the release of radioactive material in case of NPP design basis accidents (DBAs). These safety systems are manipulated by the instrumentation and control (I&C) systems which provide the control and monitoring functions of the control components that are essential for the safe operation of NPP. Recently, existing circuit-based hardware I&C systems are being replaced with microprocessor-based digital I&C systems due to analog systems approaching obsolescence and to functional advantages of digital systems [1]. Compared to the analog I&C systems, the digital systems provide advanced performance in terms of accuracy and computational capabilities and have potential for improved capabilities such as fault tolerance and diagnostics [2]. However, the use of microprocessor-based digital systems in NPP safety I&C systems triggered a big challenge in incorporating the risk characteristics of digital systems into the probabilistic risk assessment (PRA) model of a NPP in order to estimate the reliability of digital system and its risk effect on the NPP safety.

The important risk issues of digital I&C systems that should be considered in the NPP PRA model were identified by Kang and Sung [3]. Among them, the estimation of software reliability was identified as one of the important factors related to NPP risk, and a sensitivity study on the digital reactor protection system (RPS) showed the relationship of system unavailability and software failure probability. A report on operating and maintenance experience described how software error caused a significant number of digital system failures during 1990–1993 [4], where 30 failures were caused by software error, compared to 9 from random component failure. Several reports also stated the

possibility of common-mode or common-cause software failure, which may lead to significant safety threat of the digital I&C system [5, 6].

In response, quantitative software reliability methods (QSRMs) such as software reliability growth model (SRGM), Bayesian belief network (BBN) model, and test-based method have been proposed and adopted in the nuclear field. The SRGM method [7] has been widely used in software engineering field to assess the software reliability by estimating the increment of reliability as a result of fault removal over time. However, SRGM method was found to be not applicable to safety-critical software [8] because of its high sensitivity in estimating the number of faults to time-to-failure data and due to the rare software failure sets in NPP safety-critical applications which is developed under a strict development and verification and validation (V&V) life cycle.

The BBN method has also been extensively applied to estimate the software reliability of the NPP safety systems [9, 10]. The method models and aggregates disparate information on the software, such as software failure data and the quality of software life cycle activities. However, the limitation of the BBN method on quantifying the software reliability includes developing a credible BBN model that requires identification of a complete and independent set of software attributes, and the qualification of experts to estimate model parameters and qualitative evidence. Due to those limitations, the uncertainty in the estimated software residual faults and failure probability from BBN model may be very large which makes it difficult to verify the low failure probability of NPP safety software [11].

The test-based approach is an another method which assess the reliability of NPP safety-critical software by employing the standard statistical methods to the results of software testing [12]. The studies relevant to the test-based approach conducted in the nuclear field are mainly divided into two testing methods: 1) black-box testing method [13-15], and 2) white-box testing method [16-18]. The black-box testing methods consider a software program as a black-box, take random samples from its input space, determine if the outputs are correct, and use the results for statistical analyses to estimate the software reliability. However, since the black-box testing methods are conducted without the knowledge on the program's internal logic or structure, the limitations of black-box testing include the limited coverage and the completeness of the test cases [19].

On the other hand, the white-box testing methods have the advantage in taking into consideration the internal structures of the software, so the tests are performed to ensure that certain parts of the software are tested correctly with full coverage. However, since the white-box testing methods intend to test all possible paths and nodes of the software, the number of tests that must be carried out for the exhaustive testing is often very large [17]; therefore, an efficient and effective software testing framework for the safety-critical software used in NPP digital I&C systems must be developed in order to prove the correctness of the software and further quantify the software reliability based on the software test results.

The aim of this study is to develop a simulation-based software test-bed for the white-box testing of the NPP safety-critical software. The test-bed is developed by emulating the microprocessor architecture of a programmable logic controller (PLC) used in NPP digital I&C system and capturing its behavior at each machine instruction line while the software executes its dedicated safety function. The effectiveness of the proposed software testing framework is demonstrated with the safety-critical trip logic software of a fully digitalized reactor protection system (IDiPS-RPS), developed under the Korea Nuclear Instrumentation & Control Systems (KNICS) project [20]. The proposed method can effectively reduce the software testing time and effort by emulating the software behavior in a machine-level given specific software input and internal states compared to the existing black-box testing which uses trajectory inputs for software testing. The test result of the safety-critical software from the suggested method can be utilized to support the software reliability quantification of the NPP digital I&C systems and applied to the NPP PRA model to analyze the effect of software failure on the digital system availability or the NPP risk.

## 2. TARGET SYSTEM: IDiPS-RPS

The IDiPS-RPS is a digitalized RPS developed for newly constructed NPPs as well as for upgrading existing analog-based RPS [20]. It has the same function as an analog RPS which automatically generates a reactor trip signal and engineered safety features actuation signals whenever there is a demand. Fig. 1 illustrates the architecture of IDiPS-RPS which consists of four redundant channels of bistable processors (BPs) and coincidence processors (CPs) for its dedicated safety functions.

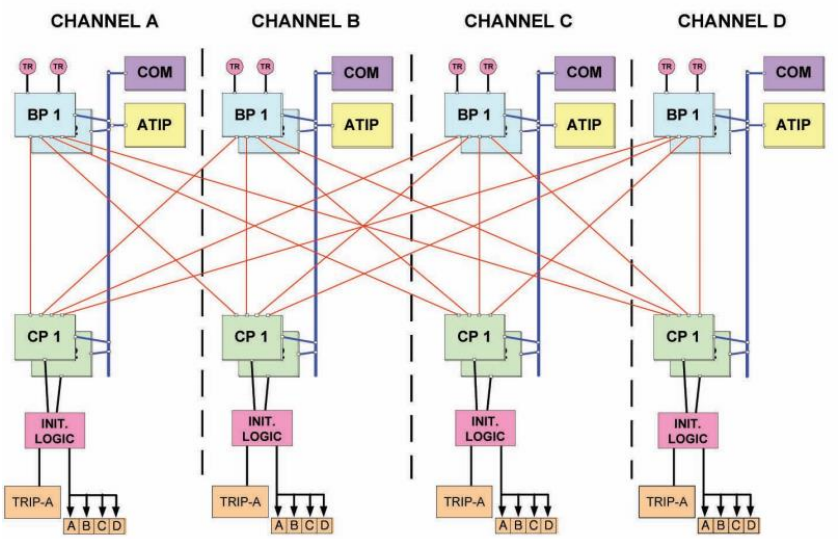


**Fig. 1. An IDiPS-RPS architecture [21]**

In each IDiPS-RPS channel, the BP determines the reactor trip state by comparing the process variables measured from the plant sensors with the predefined pre-trip or trip set-points. The CP generates a final hardware-actuating pre-trip or trip signal by conducting voting logic with the pre-trip or trip signals transferred from BPs in all channels. The processors are configured with the safety grade PLC platform (POSAFE-Q) [20] and the safety signal generation logic in each processor is implemented as a software in the PLC platform.

The POSAFE-Q consists of various modules such as a processor module, communication module, and I/O module [22]. The processor module consists of a TI C32 digital signal processor (DSP) CPU [23] and various types of memories. The application software programs implemented in each processor such as BP trip logic software and CP voting logic software are downloaded in the memory embedded within the processor module. The application software is developed in the form of function block diagram and ladder diagram (FBD/LD) programming language [24]. In the implementation phase of software lifecycle, the FBD/LD programs are compiled to machine instruction codes which can be loaded to the PLC memory area and executed by the PLC microprocessor [25].

## 3. SOFTWARE TEST-BED DEVELOPMENT

The most fundamental characteristic of PLC operation is their cyclic operation mode [26]. Each iteration of the cyclic operation of the PLC, called a scan cycle, consists of several operation stages that are sequentially repeated. After checking its own status, the PLC copies all the software input values into the RAM where variable data and user programs are stored. Then, the PLC executes the application program implemented in the memory map and the software output is updated based on the result of program execution. In each PLC scan cycle, the above operations are repeated at a fixed interval of time called a scan time.

In this study, in order to simulate the software behavior given the states of software input and internal variables, a software test-bed is developed which captures both the internal (CPU and memory

architecture) and external (the states of program input and output variables) aspects of the PLC scan cycle. The software test-bed can be used to check whether the correct output is generated by the software program using the test cases provided by the software tester. Fig. 2 shows an overview of the developed test-bed structure. The test-bed is composed of four major modules and the description of each module is as follows:

1) Architecture Module: The components of the safety grade PLC microprocessor consist of CPU register files such as 40-bit extended registers, 32-bit auxiliary registers and other registers and the memory units that is accessible to the CPU which contains the total memory space of 16Mbyte 32-bit words. Within the 16Mbyte word address space, the program, data, and I/O space are contained, allowing the program code or data of the user application software to be stored in the memory map. In the test-bed, the major components of the microprocessor are emulated in the test-bed to capture the state of CPU instruction line of application software. In order to simulate reading or writing the values from/to memory space, several different memory addressing modes including the register, direct, indirect, immediate addressing modes are implemented.
2) Assembler Module: The instruction set of safety grade PLC microprocessor contains a total of 113 instructions. All instructions are defined as a single machine word long (32-bit) and most instructions require one cycle to be executed. The categories of instruction sets include the instructions for load and store, 2-/3-operand arithmetic, program control, interlocked, and parallel operations. The syntax of each instruction set contains its specific 9-bit opcode, the addressing mode, and operands. To emulate the execution of application software in the 32-bit binary format, the functions and syntaxes of instruction sets are implemented in the test-bed.
3) Emulation Module: Based on the instruction set decoded from binary program file by the Assembler module, the operands of the instruction set from the register files are read and performs its specific operation. The operation result is written to the CPU registers or to specific memory element depending on the operands and the addressing modes. The CPU contexts such as the system stack, the condition flags stored in the CPU status register, and the data in memory area are updated at every machine instruction line.
4) Interface Module: The Interface module provides an interface between each module. For example, the instruction set decoded from the Assembler module is transferred to the to the Emulation module to conduct its specific operation. In addition, the result of instruction set execution by the Emulation module is updated to the CPU register and the memory elements emulated in the Architecture module.

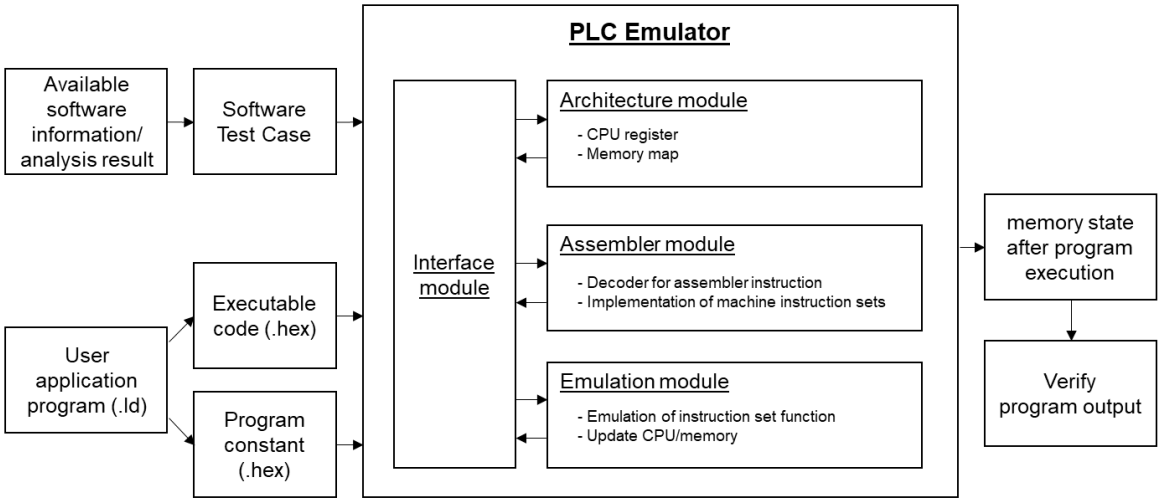


**Fig. 2. An overview of the simulation-based test-bed for safety-critical PLC software testing**

The test-bed is executed based on three basic operation processes of the PLC microprocessor: 1) fetch, 2) decode, and 3) execute [27]. In the fetch phase, the executable code which is uploaded to the memory map is fetched from the Architecture module. In the decode phase, the fetched executable code in a binary form is decoded into specific instruction set by the Assembler module. In the execute phase, the operation of the decoded instruction set is performed by the Emulation module and the operation results are stored in the CPU register or the memory. If necessary, the registers which represents the status of the microprocessor are updated during the execute phase.

When the software testing is conducted using the developed test-bed, the executable code of the safety software which was compiled from the FBD/LD language and the constant files which contain the memory map of the input (e.g. pressure, water level in NPP) and the internal variable (e.g. counter, test parameters) used in the application software are loaded to the test-bed. Then, the software test cases are uploaded to the memory area emulated in the Architecture module. After all machine instruction lines of safety software are executed, the final status of CPU registers and memory map is saved as an output file for every software test cases. The generated output files can be used to verify whether correct output is generated given the test case by checking the specific memory area that corresponds to the dedicated safety function of the application program such as trip signal.

## 4. CASE STUDY

As a case study, the proposed software testing method was applied to a KNICS IDiPS-RPS BP trip logic software. The test cases were developed based on the operational profile of the software input and internal variables. For each test case, the test results are generated by capturing the final state of output variable after the software program is executed in the developed test-bed.

### 4.1. Target Safety Software: BP pressurizer-pressure-low trip logic

In the IDiPS-RPS system, the BP compares the process variables which are transmitted from the measurement sensors in NPP with the pre-defined trip set-points. The function of each module is implemented as a software logic in a binary format in the PLC memory map. Among the BP software modules, 19 modules of trip logics are defined and the process variable of each trip logic module is compared against its pre-defined trip set-point values [28].

Among 19 trip logics, the pressurizer-pressure-low trip logic which has a variable trip set-point and operator bypass function was chosen as a case study to demonstrate the effectiveness of the proposed software test method. The pressurizer-pressure-low trip logic is one of the most complicated logics among BP trip logic modules which include various functions such as operator bypass, reset delay timer and the set-point reset by the operator. Fig. 3 shows the operation logic of the pressurizer-pressure-low trip [29].

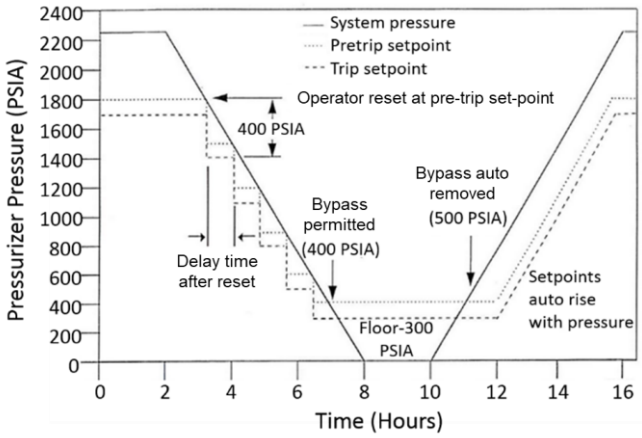


**Fig. 3. Logic flow of the IDiPS-RPS BP pressurizer-pressure-low trip logic**

The process variables of pressurizer-pressure-low trip logic which is the pressurizer pressure from the measuring instruments (0 ~ 3,000 psi) are processed to analog signals in voltage (0~10 VDC) that are converted into digital signals (0~30,000 counts) by a 15-bit analog-digital-converter [30]. The trip logic generates a trip signal if the process variable decreases below the trip set-point. When the plant is in full power mode, the trip set-point is fixed to 1779 psi. The trip set-point ranges between 1779 psi and 300 psi during shut-down and start-up processes. The operator should manually decrease the trip set-point while the pressure slowly decreases during the plant shut-down phase. When the pre-trip alarm occurs, where the pre-trip set-point is at 70 psi above the trip set-point, the operator has to push the reset button after which the trip set-point decreases 400 psi below the current pressure. Further decrease of the trip set-point is not permitted within some delay time, and bypass is permitted under 400 psi. When the pressurizer pressure increases as the plant starts up, the trip set-point is automatically set to 400 psi below the current pressure, where the trip set-point reset bypass is canceled from 500 psi.

### 4.2. Test Case Generation of Target Safety Software

From the viewpoint of NPP safety, the software testing of the BP trip logic needs to focus on the failure of its dedicated safety function, that is the on-demand failure of trip signal generation. In this study, the test cases which include the states of input and internal variables that cover all possible safety signal demand situation of a target software were developed based on the profile of each pressurizer-pressure-low trip logic variable which encountered in an actual use during plant operation. The input variables represent the software input from various sources such as the pressure or temperature signals from the measurement instruments in a NPP, the operator action from main control room or from remote shutdown room, and error signals from other modules. In this study, the profile of the process variable which is the pressurizer pressure for pressurizer-pressure-low trip was obtained from the plant thermo-hydraulic simulation. As a representative pressure transient accident, a loss of coolant accident (LOCA) was selected for trip demand condition. The plant model of APR-1400 was used for estimating the plant responses using the MARS code [31] which was developed in KAERI for a thermo-hydraulic analysis of a NPP DBAs and transient situations. As the design requirement of the IDiPS-RPS limits the scan time to less than 50 ms, the pressure deviation during the time interval between operational scan modes (100 ms) at the point of trip demand was derived based on the simulation result. In order to derive the conservative test cases for trip signal generation by the trip logic software, the plant simulation result for the hypothetical double-ended guillotine break accident case was used to derive the profile of the process variable. It is notable that the large LOCA is one of the fastest transients among the possible deviations of NPPs and the pressurizer pressure is one of the fastest process parameters.

Table 1 shows the $D_{max}$, that is the maximum digital value below trip set-point given 15-bit ADC resolution obtained from plant simulation results for various LOCA groups. In case of trip bypass request and permission variables which are inputs from the operators, they are Boolean type variables, therefore, they can have the value of either true or false. If an operator gives a trip bypass order, the system should not generate a trip signal, so only the combination of those variables which do not bypass the trip signal were examined in this study.

**Table 1: $D_{max}$ of the Pressurizer Pressure for Various LOCA Groups**

| ID | Hole diameter (inch) | $D_{max}$ (count) |
|---|---|---|
| 1 | 30 x 2* | 51 |
| 2 | 30 | 48 |
| 3 | 20 | 46 |
| 4 | 15 | 44 |
| 5 | 8 | 29 |
| 6 | 6 | 21 |

* The scenario assumes that the 30-inch diameter cold leg pipe used in reactor coolant system undergoes a double-ended guillotine break (30-inch x 2) [32]

In case of software internal variables, as the states of internal variables represent a certain state of running the software, the range of each internal variable which generates the trip signal was identified by inspecting the software logic and the other available information such as software requirement specification (SRS) or software design specification (SDS) documents [30].

Based on the obtained profile of each input and internal variable, the test cases are generated as a combination of the profile of each variable that will generate the trip signal output by the software as true. In result, a total of 705,892,684 test cases were derived. The test cases were used as an input to the developed software test-bed to verify whether the output variable updated by the BP trip logic software in the memory area matches with the expected output.

### 4.3. Test Procedure and Results of Target Safety Software

Based on the test cases derived from the possible states of each software input and internal variable as described in previous section, the test starts with initializing the software test-bed which includes emulating the CPU registers and memory elements of target digital processor. Then, the binary files of the pressurizer-pressure-low trip logic software program file which consists of the 32-bit long binary code generated from the user application program written in FBD/LD programming language and the constant file which stores the memory map of the variables used in program are loaded to the test-bed. After reading the binary file of target software, the test case file which includes the memory address and the values of software input and internal variables that should be tested is loaded in test-bed and overwrites the values in the emulated memory map. The binary program file is then executed by the test-bed until the end of the program and the value of memory address where the output variable (trip signal) is saved as an output file to check whether the software output is same with the expected output. Since the test cases are developed focusing on the trip initiation condition by the target software, the test case is verified as an error-free portion and saved as correct output if the value of the trip variable corresponds to true. However, if there is any test case that results in the value of trip signal variable as false, it is saved as wrong output which should be reviewed and debugged, and the test should be restarted from the beginning, if necessary. Fig. 4 illustrates the procedure of software testing using the developed software test-bed with the test cases.

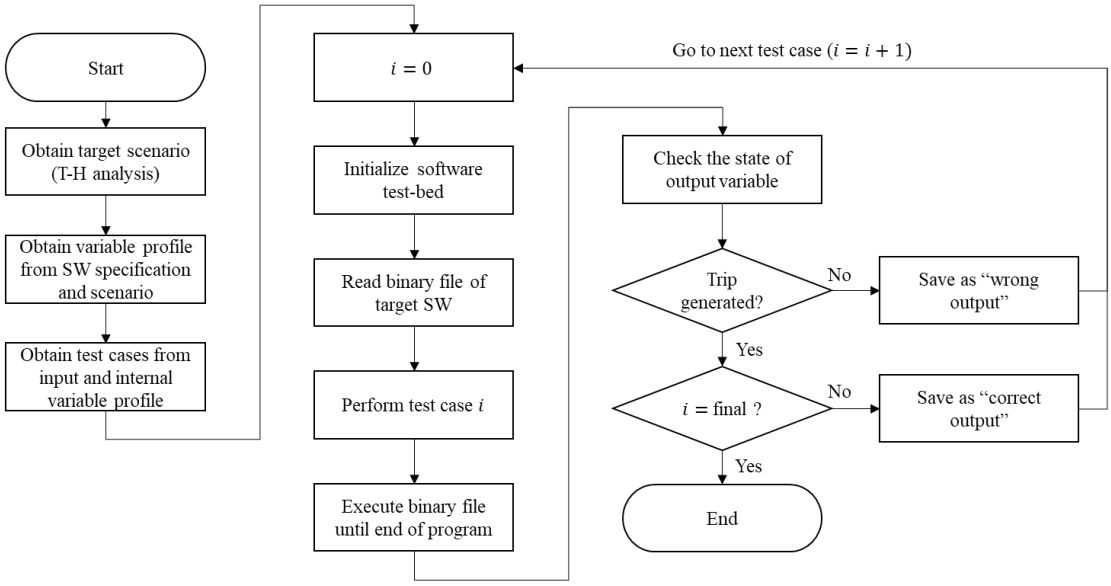


**Fig. 4. Software Testing Procedure using the Simulation-based Software Test-bed**

The BP trip logic software consists of 32,566 lines of machine instruction lines and 98,755 lines were executed in average for a single test case. Among the executed instruction sets, LDIU (load integer

unconditionally) and LDI (load integer) machine instructions were executed most frequently, 44,731 and 14,666 times, respectively. It was observed that 50.32% and 8.9% of the total execution time were spent by the LDIU and LDI instructions where the internal CPU clock used per instruction were 303 and 163 clocks in the developed software test-bed, respectively.

Fig. 5 shows a part of test results using the test cases developed for the trip initiation condition of the pressurizer-pressure-low trip logic software as a case study. The output variable of the BP trip logic software is TRIP_R_a variable which is sent to CP as a trip signal for the voting logic. The TRIP_R_a variable is packed with trip signal output of various trip logics. For example, the _6_TRIP_R variable which is the trip signal for pressurizer-pressure-low trip logic is packed at 5th bit of the TRIP_R_a variable. As shown in Fig. 5, the test result showed that the state of the TRIP_R_a variable after program execution is set to 0x20 which indicates that the 5th bit of the trip signal (PZR_PR_LO trip signal) is set to 0x1 meaning that the software generated correct output for given test case. In result, all 705,892,684 test cases developed from the previous section generated the trip signal. The test was conducted in 76.04 hours using sixteen 3.60 GHz logical processors, that is 6.205 ms were spent per test case in average using the software test-bed.

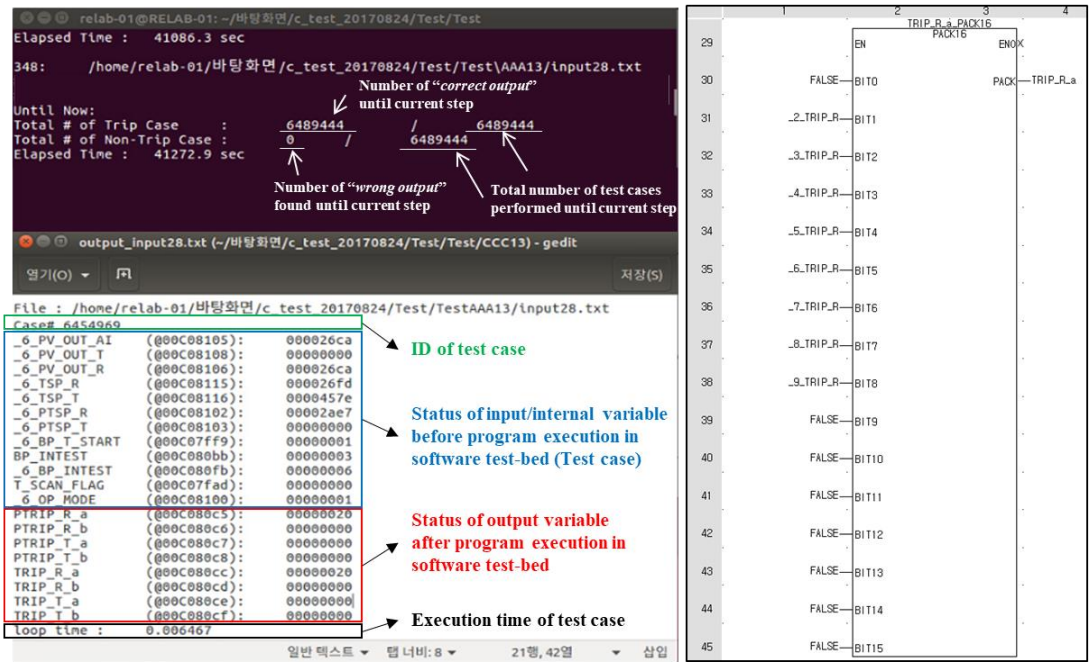


**Fig. 5. A part of test result for IDiPS-RPS BP pressurizer-pressure-low trip logic software**

## 5. SOFTWARE FAILURE PROBABILITY QUANTIFICATION FROM TEST RESULTS

A software failure probability can be quantified from the software testing results based on the test-based methods which employs standard statistical methods to the test results [12]. The test cases can be assumed to represent random samples from the software functional profile as described in previous studies [16, 33], where the probability of test cases are different as the probabilities that the actual software operational environment represented by the test case will occur are different. The total failure probability of a software can be expressed as the weighted sum of each partitioned sampling space as Equations 1 and 2 where $\hat{\theta}_t$ is the estimated software failure probability, $\hat{\theta}_i$ is the software failure probability for test case *i*, and $p_i$ is the operational profile (probability) of each test case.

$$\hat{\theta}_t = \sum_{i=1}^{n} \hat{\theta}_i \, p_i \tag{1}$$

$$\hat{\theta}_i = \begin{cases} 1 & ; \text{if test case 'i' generated correct output} \\ 0 & ; \text{if test case 'i' generated wrong output} \end{cases} \tag{2}$$

In this study, the probability of each variable having specific state was investigated based on various information such as plant dynamics [31], PLC module failure data [35], software specifications [30], and other sources. Figure 6 shows an example of the developed profile for pressurizer pressure based on thermo-hydraulic analysis of various LOCA sizes. Based on the implicit profiles of software variables, an explicit profile for each SW test set which represents the operational profile of NPP SW in an actual use was constructed.

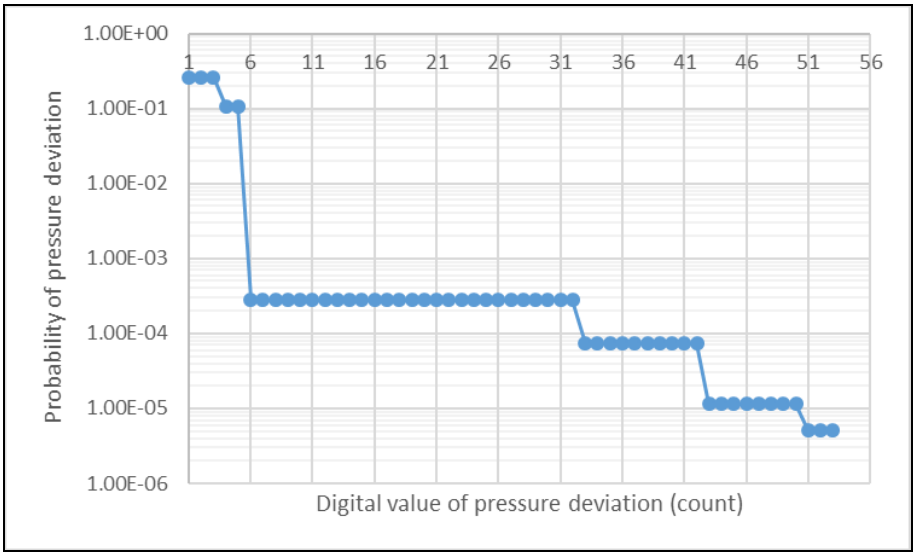


**Fig. 6. The developed input profile of pressurizer pressure deviation at a single scan interval**

Based on the derived explicit profile for the exhaustive SW test set, the number of software test set to be conducted to assure low SW probability of failure on demand (PFD) can be quantitatively derived considering that the most probable test set is tested for the effective test. By conservatively assuming the failure probabilities of the untested inputs as 1, the software failure probability after the success of the first test can be estimated as Equation 3. As one continues the test, more portions will be proven to be error-free and accomplish lower software failure probability. Table 2 shows the change of the estimated software failure probability as the test progresses for the case study conducted in this study. The estimated software PFD from the software test result can be reflected in the PRA model of a digitalized NPP and used to evaluate the risk effect of software failure on the NPP risk.

$$\hat{\theta}_t = \sum_{i=2}^{n} \hat{\theta}_i \, p_i = \sum_{i=2}^{n} p_i = 1 - p_1 \tag{3}$$

**Table 2: Software failure probabilities for case study**

| Number of test set | Explicit profile of test set ($p_i$) | SW failure probability ($\hat{\theta}_t$) |
|---|---|---|
| 1 | 1.54E-04 | 9.39E-01 |
| 2 | 1.54E-04 | 8.79E-01 |
| 3 | 1.54E-04 | 8.18E-01 |
| 4 | 1.54E-04 | 7.58E-01 |
| 5 | 1.54E-04 | 6.97E-01 |
| … | … | … |
| 75405 | 6.16E-11 | 1.00E-04 |
| 75406 | 6.16E-11 | 9.99E-05 |
| … | … | … |
| 246553 | 2.16E-13 | 1.00E-05 |
| 246554 | 2.16E-13 | 9.99E-06 |
| … | … | … |

## 6. CONCLUSION

In this study, the software test method utilizing simulation-based software test-bed was proposed. The software test-bed was developed considering the characteristics of the safety-critical PLC and the CPU architecture and memory map of the PLC microprocessor. The software test inputs for the safety-critical application such as RPS of a NPP are the inputs which cause the activation of protective action, such as a reactor trip. Therefore, the software test case was developed in consideration of the digital signal processing features of the PLC as well as the plant thermo-hydraulics data for the plant transients or accidents in a NPP. As an application of the proposed software test method, the software test cases were developed for the pressurizer-low-pressure trip of IDiPS-RPS BP software logic and tested by capturing the state of output variable stored in the memory map after the end of the trip logic program.

An important characteristic of the proposed software test approach is that the developed software test-bed can effectively reduce the software testing time per test case by emulating the software behavior given the software input and internal states in machine language level compared to the existing black-box testing. In addition, the number of test cases to achieve exhaustive testing of the safety-critical software can be quantitatively derived. Therefore, the proposed software test method can be used to support the software reliability quantification of NPP safety-critical I&C applications and further ensure the safety of the software-based digital systems.

Although the proposed framework focuses on verifying the software logic to be error-free when the demand comes, other causes of software errors should be investigated in consideration of its running environment. For example, the environment on which the software is running includes the interaction with the operating system and the hardware module. While the application software can be tested to be error-free using the proposed framework in this study, the software will not generate the safety signal if the operating system kernel does not properly call the application software or if there is any error in the hardware module that affects the application or operating system software. The external causes of potential software errors such as wrong input by the operator's mistake or the noise from the sensors or the signal transmission path also need to be considered in the future work to completely model the software failure in a NPP PRA model.

**Acknowledgements**

This work was supported by the project of 'Evaluation of human error probabilities and safety software reliabilities in digital environment (L16S092000),' which was funded by the Central Research Institute (CRI) of the Korea Hydro and Nuclear Power (KHNP) company.

**References**

[1] M. Hassan, W.E. Vesely. "*Digital I&C systems in nuclear power plants: risk-screening of environmental stressors and a comparison of hardware unavailability with an existing analog system*," Brookhaven National Laboratory, NUREG/CR-6579, 1998.
[2] National Research Council. "*Digital instrumentation and control systems in nuclear power plants: safety and reliability issues*," National Academies Press, 1997.
[3] H. G. Kang, T. Sung. "*An analysis of safety-critical digital systems for risk-informed design*," Reliability Engineering & System Safety, 78, pp. 307-314, (2002).
[4] H. Ragheb. "*Operating and maintenance experience with computer-based systems in nuclear power plants*," International Workshop on Technical Support for Licensing Issues of Computer-Based Systems Important to Safety, München, Germany, (1996).
[5] U.S. Nuclear Regulatory Commission. "*Guidance for Evaluation of D3 in Digital Computer-Based Instrumentation and Control Systems*," U.S. Nuclear Regulatory Commission, BTP 7-19-Rev. 6, 2012.

[6] K. Korsah, M. D. Muhlheim, R. Wood. "*A Qualitative Assessment of Current CCF Guidance Based on a Review of Safety System Digital Implementation Changes with Evolving Technology*," Oak Ridge National Laboratory, ORNL/SR-2016/148, 2016.
[7] M. R. Lyu. "*Handbook of software reliability engineering*," McGraw-Hill, 1996, New York.
[8] M. C. Kim, S. C. Jang, J. Ha. "*Possibilities and limitations of applying software reliability growth models to safety critical software*," Nuclear Engineering and Technology, 39, pp. 145-148, (2007).
[9] N. Fenton, M. Neil, W. Marsh, P. Hearty, D. Marquez, P. Krause, R. Mishra. "*Predicting software defects in varying development lifecycles using Bayesian nets*," Information and Software Technology, 49, pp. 32-43, (2007).
[10] H. S. Eom, G. Y. Park, S. C. 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, 51, pp. 38-49, (2013).
[11] S. Brown. "*Overview of IEC 61508. Design of electrical/electronic/programmable electronic safety-related systems*," Computing & Control Engineering Journal, 11, pp. 6-12, (2000).
[12] T. L. Chu, M. Yue, G. Martinez-Guridi, J. Lehner. "*Review of quantitative software reliability methods*," Brookhaven National Laboratory, BNL-94047-2010, 2010.
[13] J. May, G. Hughes, A. D. Lunn. "*Reliability estimation from appropriate testing of plant protection software*," Software Engineering Journal, 10, pp. 206-218, (1995).
[14] 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. Nuclear Regulatory Commission, NUREG/CR-7044, 2013.
[15] S. Kuball, J. H. R. May. "*A discussion of statistical testing on a safety-related application*," Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 221, pp. 121-132, (2007).
[16] 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, 94, pp. 1542-1546, (2009).
[17] S. M. Shin, S. H. Lee, H. G. Kang, H. S. Son, S. J. Lee. "*Test based reliability quantification method for a safety critical software using finite test sets*," Proceedings of the 9th International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human–Machine Interface Technologies (NPIC & HMIT 2015), Charlotte, NC, (2015).
[18] S. M. Shin, J. Cho, W. Jung, S. J. Lee. "*Test based reliability assessment method for a safety critical software in reactor protection system*," Proceedings of the 9th International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human–Machine Interface Technologies (NPIC&HMIT 2017), San Francisco, CA, (2017).
[19] C. V. Ramamoorthy, W. T. Tsai. "*Advances in software engineering, Computer*," 29, pp. 47-58, (1996).
[20] K. C. Kwon, M. S. Lee. "*Technical review on the localized digital instrumentation and control systems*," Nuclear Engineering and Technology, 41, pp. 447-454, (2009).
[21] J. G. Choi, S. J. Lee, H. G. Kang, S. Hur, Y. J. Lee, S. C. Jang. "*Fault detection coverage quantification of automatic test functions of digital I&C system in NPPS*," Nuclear Engineering and Technology, 44, pp. 421-428, (2012).
[22] M. Lee, S. Song, D. Yun. "*Development and application of POSAFE-Q PLC platform*," International Atomic Energy Agency, IAEA-CN-194, 2012.
[23] Texas Instruments. "*TMS320C3x User's guide*," Texas Instruments, 1997.
[24] International Electrotechnical Commission. "*Programmable Controllers - Part 3: Programming Languages*," International Electrotechnical Commission, IEC 61131-3, 1993.
[25] K. Koo, B. You, T. W. Kim, S. Cho, J. S. Lee. "*Development of application programming tool for safety grade PLC (POSAFE-Q)*," Transactions of the Korean Nuclear Society Spring Meeting, Chuncheon, Korea, (2006).
[26] J. Palomar, R. H. Wyman. "*The programmable logic controller and its application in nuclear reactor systems*," U.S. Nuclear Regulatory Commission, NUREG/CR-6090, 1993.
[27] W. Bolton. "*Programmable logic controllers*," Newnes, 2015.

[28] J. Yoo, J. H. Lee, J. S. Lee. "*A research on seamless platform change of reactor protection system from PLC to FPGA*," Nuclear Engineering and Technology, 45, pp. 477-488, (2013).
[29] J. G. Choi, D. Y. Lee. "*Development of RPS trip logic based on PLD technology*," Nuclear Engineering and Technology, 44, pp. 697-708, (2012).
[30] Y. H. Koo, S. H. Lim, S. J. Lee. "*BP SDS for Reactor Protection System*," Doosan Heavy Industries and Construction Co., Ltd, KNICS-RPS-SDS231-Rev. 3, 2008.
[31] J. J. Jeong, K. S. Ha, B. D. Chung, W. J. Lee. "*Development of a multi-dimensional thermal-hydraulic system code - MARS 1.3.1*," Annals of Nuclear Energy, 26, pp. 1611-1642, (1999).
[32] The Other Dynamic Loads and Load Combinations Task Group. "*Report of the US nuclear regulatory commission piping review committee*," U.S. Nuclear Regulatory Commission, NUREG/1061, 1984.
[33] K. W. Miller, L. J. Morell, R. E. Noonan, S. K. Park, D. M. Nicol, B. W. Murrill, M. Voas. "*Estimating the probability of failure when testing reveals no failures*," IEEE transactions on Software Engineering, 18(1), pp. 33-43 (1992).
[34] S. A. Eide, D. M. Rasmuson, C. L. Atwood, "*Estimating Loss-of-Coolant Accident Frequencies for the Standardized Plant Analysis Risk Models*," Idaho National Laboratory (INL), INL/CON-08-13778, 2008.
[35] J. G. Choi, "*Reliability Analysis Report of Safety Grade Programmable Logic Controller (POSAFE-Q)*," Korea Atomic Energy Research Institute (KAERI), KNICS-PLC-AR103, Rev.01, 2007.