DEVELOPMENT OF SIMULATION-BASED TESTING ENVIRONMENT FOR SAFETY-CRITICAL SOFTWARE IN DIGITALIZED NUCLEAR POWER PLANT

Sang Hun Lee and Hyun Gook Kang*
Department of Mechanical Aerospace and Nuclear Engineering
Rensselaer Polytechnic Institute (RPI)
110 8th street, Troy, NY, USA, 12180
lees35@rpi.edu; kangh6@rpi.edu

Seung Jun Lee
School of Mechanical, Aerospace and Nuclear Engineering
Ulsan National Institute of Science and Technology (UNIST)
50 UNIST-gil, Ulsan, Republic of Korea, 44919
sjlee42@unist.ac.kr

Sung Min Shin
Korea Atomic Energy Research Institute (KAERI)
111 Daedeok-daero, 989beon-gil, Yuseong-gu, Daejeon, Republic of Korea, 34057
smshin@kaeri.re.kr

Eun-chan Lee
Korea Hydro & Nuclear Power Co., Ltd.,
1655 Bulguk-ro, Gyeongju-si, Gyeongsangbuk-do, Republic of Korea, 38120
eclee5639@khnp.co.kr

ABSTRACT

An issue on incorporating the software reliability within the NPP PRA model has been emerged in the licensing processes of digitalized NPPs. Since software failure induces CCFs 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 order to prove the software to be error-free or have very low failure probability, an exhaustive testing of software is required. In this study, a software testing method based on the MCS-based exhaustive test case generation scheme combined with the simulation-based test-bed is proposed. The software test-bed was developed by emulating the microprocessor architecture of PLC used in NPP safety-critical applications and capturing its behavior at each machine instruction. For the test case generation, the software logic model was developed from the formal definition of FBD/LD and the sets of MCSs which represent the necessary and sufficient conditions for the software variables' states to produce safety software output were generated. The MCSs were then converted into the test sets which are used as inputs to test-bed to verify that the test cases produce correct output after software execution. The effectiveness of the proposed method is demonstrated with the safety-critical trip logic software of IDiPS-RPS, a fully digitalized reactor protection system. The method provides a systematic way to conduct software exhaustive testing while effectively reducing the software testing effort by emulating PLC behavior in machine-level compared to existing software testing methods.

Key Words: Nuclear Power Plant; Digital I&C System; Reactor Protection System; Safety-critical Software; Software Testing

1 INTRODUCTION

The safety systems of nuclear power plant (NPP) are designed to protect the public from the release of radioactive material in case of NPP design basis accidents (DBAs), thus are manipulated by the instrumentation and control (I&C) systems which provide the control and monitoring functions of the critical components that are essential for the safe operation of NPP. As analog I&C systems approaches to obsolescence, the circuit-based hardware I&C systems are being replaced with microprocessor-based digital I&C 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, self-testing and diagnostics [2]. However, the use of microprocessor-based digital I&C systems triggered a challenge in incorporating the risk of digital systems into the NPP probabilistic risk assessment (PRA) model and in estimating the digital system reliability and its risk effect on the NPP safety. The risk factors of digital I&C systems were identified by Kang and Sung [3] and the risk issues related to software such as software reliability [3] and common-cause failure (CCF) [5, 6] were identified as important factors contributing to NPP risk. In addition, 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.

In response, various quantitative software reliability methods (QSRMs) such as software reliability growth model (SRGM) [7, 8], Bayesian belief network (BBN) model [9-11], and test-based method [12-15] have been proposed and adopted in the nuclear field. Among them, the test-based approach is a method which assess the reliability of NPP safety-critical software by employing the standard statistical methods to the results of software testing and is mainly divided into two testing methods: 1) black-box testing method [12, 13], and 2) white-box testing method [14, 15]. The black-box testing method considers a software program as a black-box, take random samples from its input space, determine whether the generated 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 of the test cases [16]. The white-box testing methods takes into account the internal structures of the software so that the tests are performed to ensure that specific parts of the software are tested correctly with full coverage. However, since the white-box testing methods intend to test all possible paths 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 must be developed in order to prove the integrity of the software and quantify the software reliability based on the software test results.

In this study, a software testing method based on the minimal cutset (MCS)-based exhaustive test case generation scheme combined with the simulation-based test-bed is proposed. From the viewpoint of NPP safety, the testing of the safety software needs to focus on the failure of its dedicated safety function that is the failure of trip signal generation when demand comes. Since the output of the safety software is determined by the combinations of the software variables' states, the exhaustive test set can be generated by deriving the necessary and sufficient conditions of the software variables' states that generates the safety signal. In this study, the software logic model where the output of the software is represented as logical expressions of software variables' states is constructed based on the formal definition of function block diagram/ladder diagram (FBD/LD). The set of MCSs for the software output is generated from the software logic model and are converted to the test set files which are used as an input to the simulation-based test bed. In the simulation-based test-bed, the test-bed captures both the internal (e.g. CPU registers and memory elements) and external (the states of program input/output variables) aspects of the programmable logic controller (PLC) execution characteristics so that the software behavior is simulated given the software test set file and check whether the correct output is generated by the safety software application program. 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. The proposed method can systematically generate the exhaustive test sets for safety-critical software while effectively reducing the software testing time and effort by emulating the execution of the software in a machine-level compared to the previous 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 NPP risk.

2 EXHAUSTIVE TEST SET GENERATION FOR NPP SAFETY SOFTWARE

From the viewpoint of NPP safety, the testing of the safety software needs to focus on the failure of its dedicated safety function, that is the on-demand failure of the safety signal generation such as reactor trip signal. FBD/LD is one of the widely used PLC programming languages defined in IEC61131-3 [18] and is usually translated into other languages such as C, Verilog, or specific machine code for PLC implementation, simulation, or verification [19]. Figure 1 shows a part of FBD for RESET_FALLING logic of a KNICS RPS bistable processor (BP) [20]. In order to generate the exhaustive software test set, the software logic model is constructed where the software logic is modeled with the Boolean representation of the software input and internal variables that contributes to generating the safety signal output. Figure 1 shows an example of the software logic model for reactor trip signal generation. The top event of the software logic model is defined as the safety signal output of the trip logic software.

<Example Software Logic Flow>

// (1) Error in the range of process variable value
If (PV_OUT < RNG_MIN) || (AI_OVERRNG_HI)
Then AI_RNG_ERR := TRUE;
Else AI_RNG_ERR := FALSE;

// (2) Pre-trip signal
If (AI_RNG_ERR || AI_MDL_ERR || AI_CH_ERR || PTRIP_LOGIC)
Then PTRIP := TRUE;
Else PTRIP := FALSE;

// (3) Trip signal
If (AI_RNG_ERR || AI_MDL_ERR || AI_CH_ERR ||TRIP_LOGIC)
Then TRIP := TRUE;
Else TRIP := FALSE;

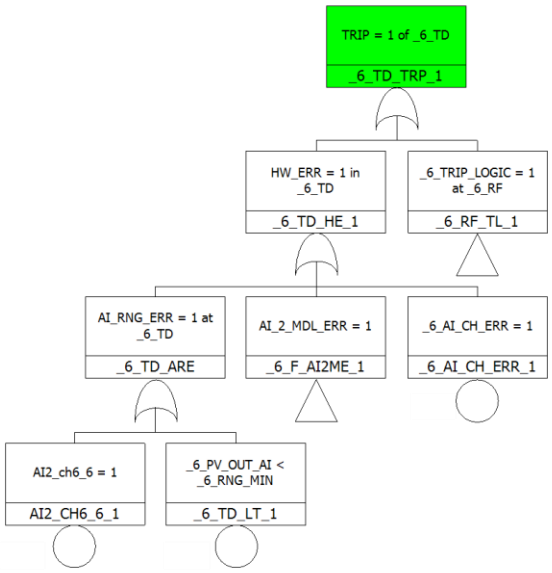


Figure 1. An example of software logic model developed based on the software specifications - (Δ) Transfer in: indicates that tree is developed further but not shown here; (○): basic event.

MCS analysis is widely used technique in reliability field for manipulating the logic structure to identify all combinations of basic events that result in the occurrence of the top event [21]. The result of MCS analysis is a new logic tree, logically equivalent to the original, consisting of OR gates beneath the top event whose inputs are the MCSs. Each MCS is composed of AND gates containing a set of basic events (e.g. component failure modes) necessary and sufficient to cause the top event (e.g. system failure mode). Similar approach can be used to derive the necessary and sufficient conditions of software

variables' states (modeled as basic events) that generate the NPP safety SW output (modeled as top event). Figure 2 shows a part of generated MCSs for the trip signal shown in Fig. 1. The software test sets are then derived from each MCS (the result of software logic model) by converting the basic events in each MCSs to all software variables' states that satisfies the basic event conditions. Figure 3 shows an overview of the proposed exhaustive test set generation framework for NPP safety software testing.

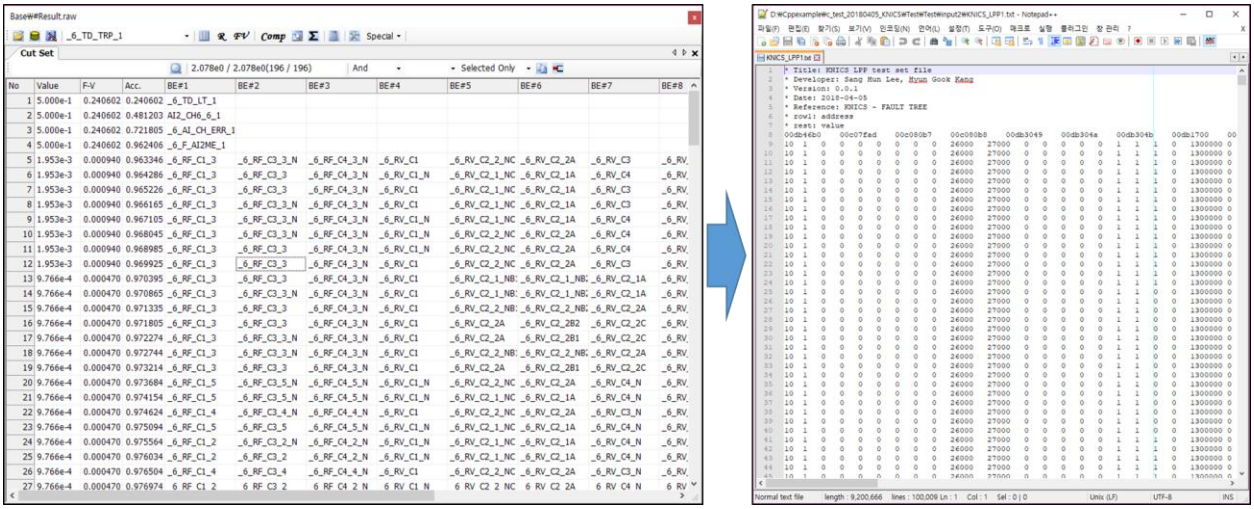


Figure 2. An example of minimal cutset generated from the software logic model.

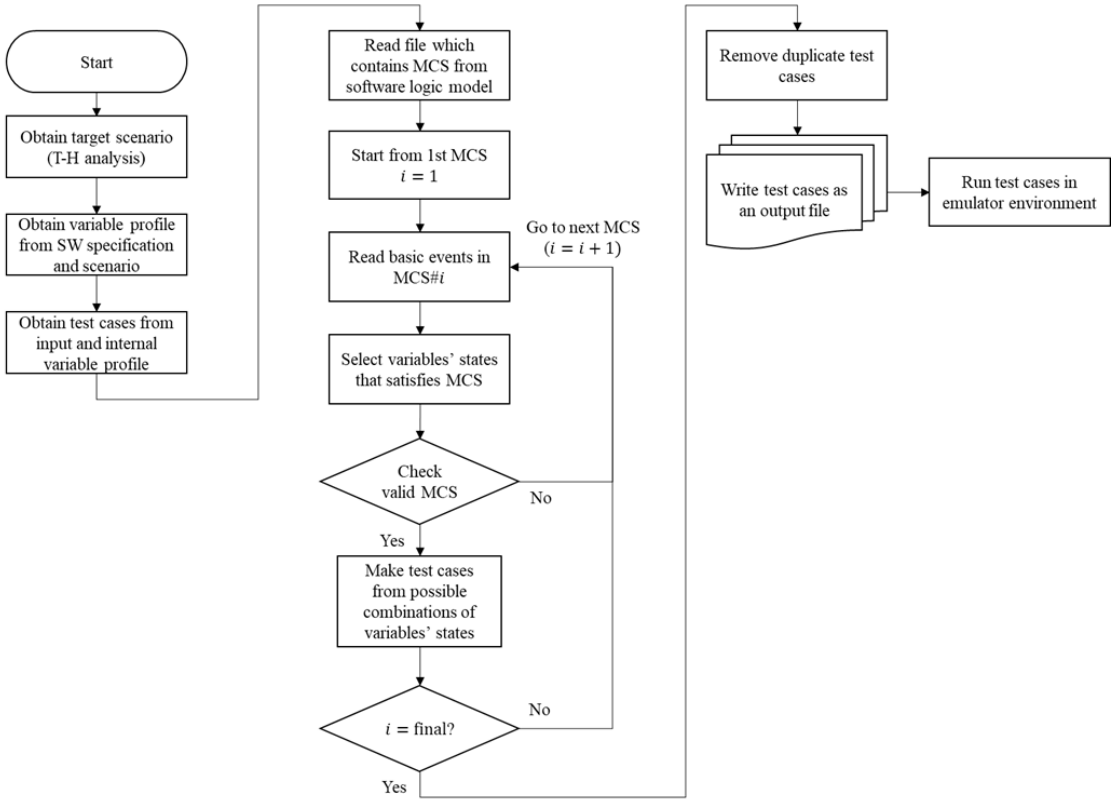


Figure 3. Proposed exhaustive test set generation framework for NPP safety software testing.

3 SIMULATION-BASED NPP SAFETY SOFTWARE TEST-BED

The PLC operation is characterized by their cyclic operation mode [22]. Each iteration of the cyclic operation of the PLC (scan cycle) consists of four operation stages that are sequentially repeated: 1) fetch, 2) decode, 3) read, and 4) execute. After checking the PLC status, the PLC copies the software input values into the random access memory (RAM) where input/internal/output variable data, user programs are stored. Then, the central processing unit (CPU) executes the loaded application program installed in the memory map and the software output is updated based on the execution result of the application program. These operations are repeated at a fixed interval of time called a scan time.

In order to simulate the software behavior given the states of software input and internal variables, a software test-bed was developed which captures both the internal (CPU and memory architecture) and external (the states of program input/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 given the test cases provided by the software tester. Figure 4 shows an overview of the developed test-bed structure.

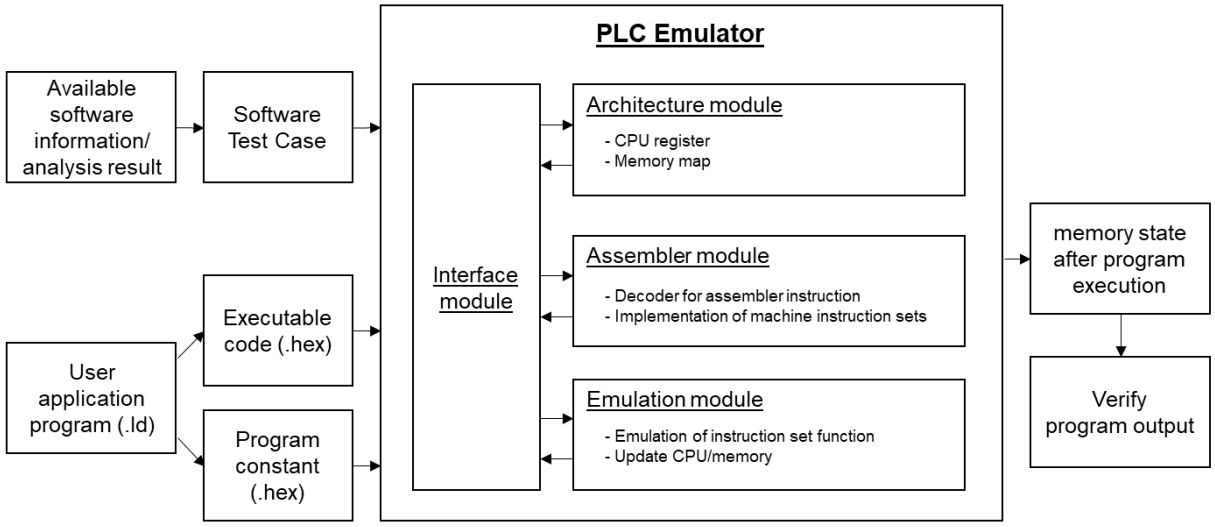


Figure 4. An overview of the simulation-based test-bed for NPP safety-critical software testing.

In the developed test-bed, as fetch phase, the executable code which is uploaded to the memory map is fetched from the *Architecture module*. As a decode phase, the fetched executable code loaded as a binary form in the test-bed is decoded into specific instruction set by the *Assembler module*. As an execute phase, the operation of the decoded instruction set is performed by the *Emulation module* where the functions of machine instruction sets of a target PLC microprocessor is emulated. During the execute phase, the test-bed updates the registers and the memory elements according to the decoded instruction.

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) and the internal variables (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 given software test sets. 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 reactor trip signal generation.

4 CASE STUDY: IDIPS-RPS

To demonstrate the feasibility of the proposed software testing method, a case study of KNICS IDiPS-RPS BP trip logic software was conducted. As a case study, the proposed software testing method was applied to a KNICS IDiPS-RPS BP trip logic software. The test sets were generated from the MCSs for the target software output (reactor trip signal) derived from the software logic model. For each test set, 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: RPS BP trip logic

The IDiPS-RPS is a digitalized reactor protection system (RPS) developed in the KNICS project for newly constructed NPPs, as well as for upgrading existing analog-based RPS [20]. It has the same function as an analog RPS to automatically generate a reactor trip signal and engineered safety features actuation signals (ESFAS) whenever demand comes from the plant monitoring instrumentations or during the automated/manual test period. In the IDiPS-RPS, the bistable processor (BP) determines the trip state by comparing the process variables (e.g. pressure, water level in NPP) with the predefined pre-trip or trip set-points and coincidence processors (CP) generates a hardware-actuating trip signal based on the voting logic of various trip signals from BPs. Each processor in the IDiPS-RPS is configured with the safety grade PLC platform (POSAFE-Q) [23] and the function of BP/CP is implemented as a software.

Among 19 trip logics in BP, the pressurizer-pressure-low trip (PZR_PR_LO) logic which has a variable trip set-point and operator bypass function was chosen as a case study. The pressurizer-pressure-low trip logic is one of the most complicated trip logics in BP and contains various functions such as operator bypass, reset delay timer and the set-point reset by the operator in case of reactor startup/shutdown. Figure 5 shows the operation logic of the pressurizer-pressure-low trip [24].

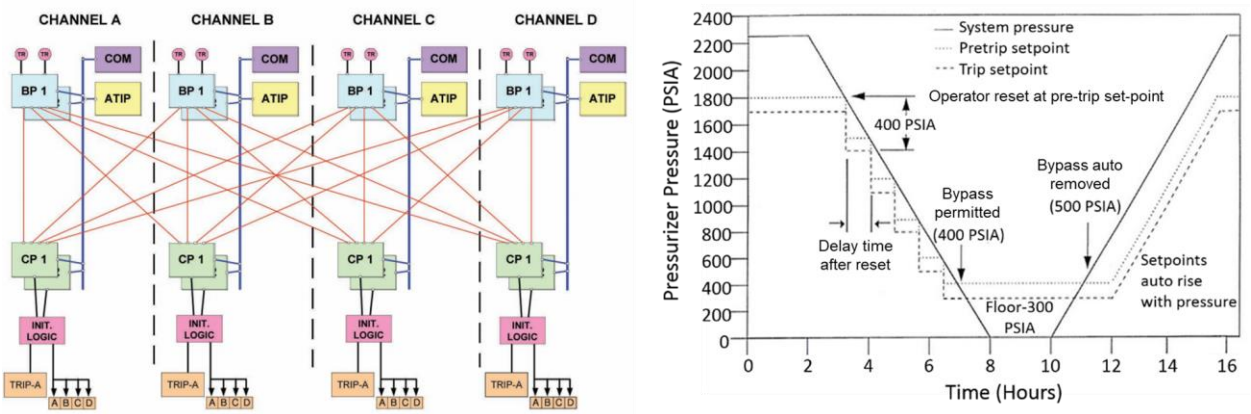


Figure 5. A block diagram of IDiPS-RPS and the pressurizer-pressure-low trip logic in BP.

4.2 Test Case Generation of Target Safety Software

Based on the proposed framework, the software logic model for the pressurizer-pressure-low trip logic is developed using Advanced Information Management System for Probabilistic Safety Assessment

(AIMS-PSA) software [25] developed by Korea Atomic Energy Research Institute (KAERI) based on the software requirement specifications (SRS) [26] and software design specification (SDS) documents [27] of KNICS RPS BP trip logic software. In generating the MCS of the software logic model, Fault Tree Reliability Evaluation eXpert (FTREX) software package [28] was used which implements coherent BDD algorithm to solve the software logic model. In result a total of 150,285 MCSs were generated which included the 138 basic events. The software variables that contributes to trip signal generation include 34 variables (including process variables, test parameters, counter and etc.) and 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 547,471,360 test cases were derived. Figure 6 shows an overview of the exhaustive test set generation procedures for the case study.

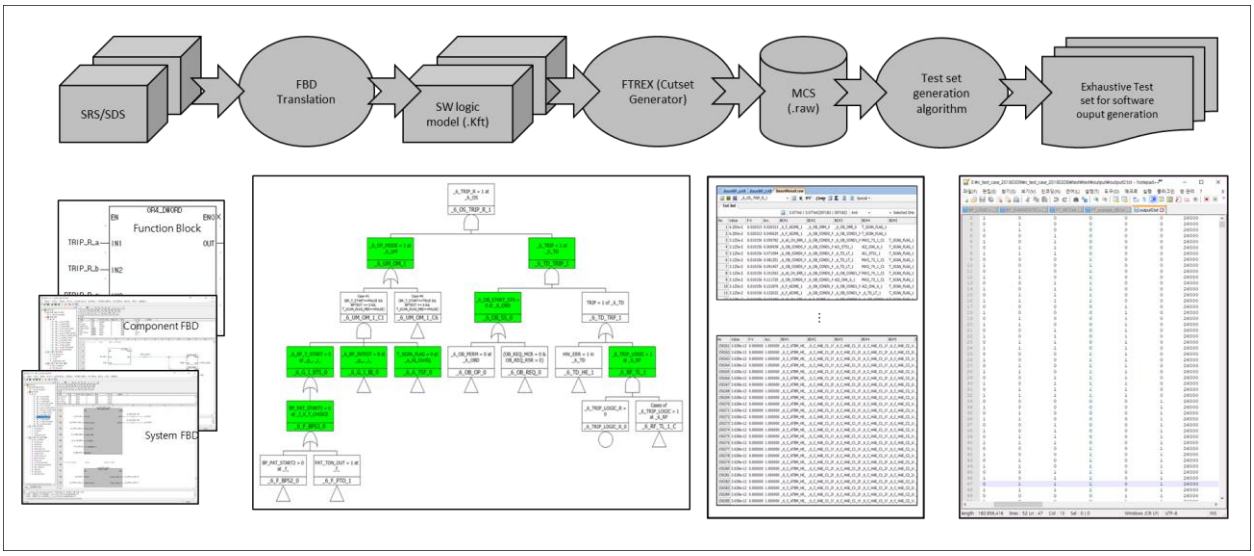
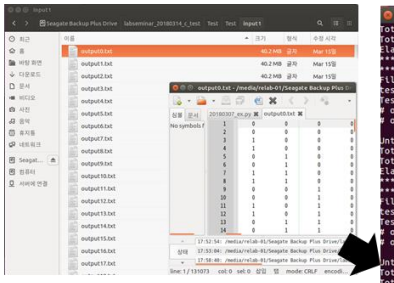


Figure 6. Test set generation scheme for KNICS RPS BP PZR_PR_LO trip logic.

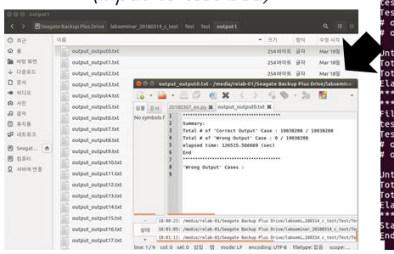
4.3 Test Case Generation of Target Safety Software

The test starts with initializing the software test-bed which includes emulating the CPU registers and memory elements of the PLC processor. The binary files of the BP trip logic software including: 1) the program file which consists of the 32-bit long binary code generated from the user application program written in FBD/LD and 2) the constant file which stores the memory map of the program variables are loaded to the test-bed. Then, the test case file derived from the previous section that includes the memory address and the values of software input/internal variables is loaded in the emulated memory map of the test-bed. The 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 (PZR_PR_LO trip signal) is saved as an output file to check whether the software output after program execution is same with the expected output (i.e. trip signal generated). 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. 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. Figure 7 shows a part of test results using the test cases developed for the trip initiation condition of the PZR_PR_LO trip logic software. 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. All the 547,471,360 test cases developed from previous section generated the trip signal for PZR_PR_LO

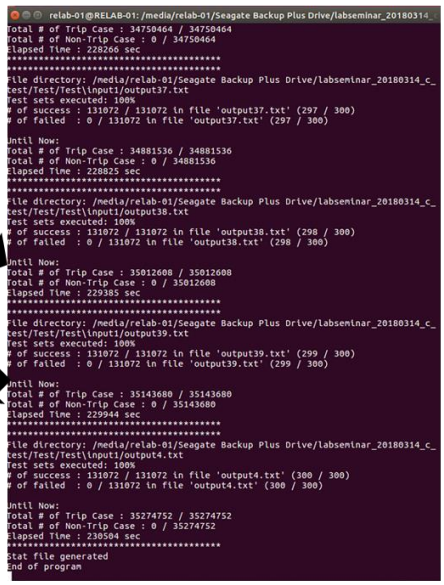
trip logic and the test was conducted in 62.03 hours using sixteen 3.60 GHz logical processors, that is 6.53 ms were spent per test case in average in the software test-bed.



Example of SW test set file (input to test-bed)



Example of SW test result file (output from test-bed)



SW Test-bed execution screen (parallel computing: output_i folder -> logical cpu #i)

cpu #	folder	cases	correct output (trip)	wrong output (non-trip)	time (sec)	
1	output1	35274752	35274752	0	229943.6555	
2	output2	34406400	34406400	0	230420.6961	
3	output3	34406400	34406400	0	224942.8522	
4	output4	34406400	34406400	0	224154.2760	
5	output5	34406400	34406400	0	220544.8760	
6	output6	34406400	34406400	0	223320.6415	
7	output7	34406400	34406400	0	220768.3832	
8	output8	34406400	34406400	0	223486.4021	
9	output9	34406400	34406400	0	226299.2078	
10	output10	34406400	34406400	0	230532.3867	
11	output11	34406400	34406400	0	221057.2509	
12	output12	34406400	34406400	0	223758.5775	
13	output13	34406400	34406400	0	220685.7890	
14	output14	34406400	34406400	0	223548.6662	
15	output15	34406400	34406400	0	230559.9196	
16	output16	30507008	30507008	0	199046.3575	
			547471360		3573069.9378	0.0065265

Statistics of SW test result

Figure 7. A part of testing result for BP PZR_PR_LO trip logic software.

5 CONCLUSION

In this study, a software testing method based on the MCS-based exhaustive test case generation scheme combined with the simulation-based test-bed is 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. Since the software test inputs for the safety-critical application such as NPP RPS are the inputs which cause the activation of protective action, such as a reactor trip, the software test case was generated by constructing software logic model developed from the FBD/LD of the target program. The MCSs generated from the software logic model were then converted to derive the exhaustive test sets which are used as inputs to test-bed to verify that the test cases produce correct output after software execution. As an application of the proposed software test method, the software test cases were developed for the PZR_LO_PR trip of KNICS 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 exhaustive SW test set which represents NPP safety SW demand can be quantitatively derived by constructing SW logic model and deriving test sets from MCSs using the proposed method. In addition, the functionality of the NPP safety SW can be proven based on the test result without uncertainties compared to the conventional black-box testing conducted in nuclear field which used test sets randomly sampled from the SW operational profile. By utilizing the simulation-based test-bed, it 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. Thus, 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 demand comes, other causes of software errors should be investigated in consideration of its running environment. The environment on which the software is running includes not only the software itself but also the interaction between the software and the operating system and the hardware module. While the application software can be tested to be error-free using the proposed framework, 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 modules that affect the 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 have to be considered to reflect the software failure in NPP PRA model.

6 ACKNOWLEDGEMENT

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.

7 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*, NUREG/CR-6579, Brookhaven National Laboratory, Upton, NY, USA (1998).
2. National Research Council, *Digital instrumentation and control systems in nuclear power plants: safety and reliability issues*, National Academies Press, Washington D.C., USA (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. NRC, *Guidance for Evaluation of D3 in Digital Computer-Based Instrumentation and Control Systems*, BTP 7-19-Rev. 6, U.S. Nuclear Regulatory Commission, Washington D.C., USA (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, ORNL/SR-2016/148, Oak Ridge National Laboratory, Oak Ridge, TN, USA (2016).
7. M. R. Lyu., *Handbook of software reliability engineering*, McGraw-Hill, NY, USA (1996).
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. H. G. Kang, S. H. Lee, S. J. Lee, T.-L. Chu, A. Varuttamaseni, M. Yue, H. S. Eom, J. Cho, M. Li, "Development of a Bayesian belief network model for software reliability quantification of digital protection systems in nuclear power plants," *Annals of Nuclear Energy*, **120**, pp. 62-73 (2018).

12. J. May, G. Hughes, A. D. Lunn, “Reliability estimation from appropriate testing of plant protection software,” *Software Engineering Journal*, **10**, pp. 206-218 (1995).
13. T. L. Chu, M. Yue, G. Martinez-Guridi, J. Lehner, *Development of quantitative software reliability models for digital protection systems of nuclear power plants*, NUREG/CR-7044, U.S. Nuclear Regulatory Commission, Washington D.C., USA (2013).
14. 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).
15. 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 10th International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human–Machine Interface Technologies (NPIC&HMIT 2017)*, San Francisco, CA, June 11-15 (2017).
16. C. V. Ramamoorthy, W. T. Tsai, “Advances in software engineering”*, Computer*, **29**, pp. 47-58 (1996).
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, February 22-26 (2015).
18. IEC, *Programmable Controllers - Part 3: Programming Languages*, IEC 61131-3, International Electrotechnical Commission, Geneva, Swiss (1993).
19. D. A. Lee, J. Yoo, J. S. Lee, “Guidelines for the Use of Function Block Diagram in Reactor Protection Systems,” *2014 21st Asia-Pacific Software Engineering Conference (APSEC)*, Jeju, Korea, December 1-4 (2014).
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. Center for Chemical Process Safety, *Guidelines for chemical process quantitative risk analysis*, Center for Chemical Process Safety/AIChE, New York, NY (2000).
22. J. Palomar, R. H. Wyman, *The programmable logic controller and its application in nuclear reactor systems*, NUREG/CR-6090, U.S. Nuclear Regulatory Commission, Washington D.C., USA (1993).
23. 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).
24. G. Y. Park, K. Y. Koh, E. Jee, P. H. Seong, K. C. Kwon, D. H. Lee, “Fault tree analysis of KNICS RPS software,” *Nuclear Engineering and Technology*, **40**, pp. 397-408 (2008).
25. S. H. Han, H. G. Lim, S. C. Jang,. “AIMS-PSA: a software for integrating various types of PSAs,” *9th Probabilistic Safety Assessment and Management (PSAM9)*, Hong Kong, China, May 18-23 (2008).
26. Doosan Heavy Industries and Construction Co., Ltd, *RPS Functional Requirement Specification*, KNICS-RPS-DS101-Rev. 02 (2006).
27. Doosan Heavy Industries and Construction Co., Ltd, *BP SDS for Reactor Protection System*, KNICS-RPS-SDS231-Rev. 03 (2008).
28. Jung, W. S., Han, S. H., Ha, J. J., 2005. “An overview of the fault tree solver FTREX,” *In 13th International Conference on Nuclear Engineering*, Beijing, China, May 16-20 (2005).