Development of Simulation-based Test-bed for Nuclear Power Plant Safety-critical Software

Sang Hun Lee[1], Seung Jun Lee[2], Jinkyun Park[3], Eun-chan Lee[4], Hyun Gook Kang[1]

[1]*Department of Mechanical, Aerospace and Nuclear Engineering, Rensselaer Polytechnic Institute (RPI), 110 8th street, Troy, NY, USA, 12180, lees35;kangh6@rpi.edu*
[2]*School of Mechanical, Aerospace and Nuclear Engineering, Ulsan National Institute of Science and Technology (UNIST), 50 UNIST-gil, Ulsan, Republic of Korea, 44919, sjlee420@unist.ac.kr*
[3]*Integrated Safety Assessment Division, Korea Atomic Energy Research Institute (KAERI), 111 Daedeok-daero, 989beon-gil, Yuseong-gu, Daejeon, Republic of Korea, 34057, kshpjk@kaeri.re.kr*
[4]*Korea Hydro & Nuclear Power Co., Ltd., 1655 Bulguk-ro, Gyeongju-si, Gyeongsangbuk-do, Republic of Korea, 38120, eclee@khnp.co.kr*

INTRODUCTION

An issue on incorporating the software reliability of digital instrumentation and control (I&C) systems in nuclear power plant (NPP) probabilistic risk assessment (PRA) emerged during the licensing process revealed the importance of the reliability quantification of the safety-critical software used in NPP safety applications [1]. Since the software failure induces the common cause failure (CCF) of processor modules in NPP digital I&C system, the risk effect of software failure on the system unavailability and plant-level risk is significant [2]. Therefore, the quantification of software reliability plays a very important role in ensuring the safety of NPP, and the verification of a very low software failure probability is crucial regarding the PRA of digitalized NPP.

In response, the quantitative software reliability methods such as software reliability growth model (SRGM) [3], Bayesian belief network (BBN) model [4, 5], and test-based method [6-9] have been proposed that was adopted in the nuclear field. However, it has been generally known that SRGMs which is one of the most mature techniques for software reliability assessment cannot be applied to safety-critical software [3]. This is mainly because of the lack of software failure sets in NPP safety-critical applications and its high sensitivity in estimating the number of faults to time-to-failure data. The BBN methods have also been extensively applied to estimate the software reliability of the NPP safety systems. However, the limitations of the BBN model include 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.

The test-based approach is an another method which employs standard statistical methods to the results of software testing to assess the software reliability [10] and is mainly divided into two categories: 1) the black-box testing method [6, 7], and 2) the white-box testing method [8, 9]. The black-box testing method takes random samples from its input space in the form of trajectory-input, determines if the outputs are correct, and uses the results for statistical analyses to estimate the software reliability. However, since the black-box testing is 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 [11]. On the other hand, the white-box testing method has the advantage in taking into consideration the internal structures of the software. The tests are performed to ensure that certain parts of the software are tested correctly with proper coverage. However, since the white-box testing methods intend to test all possible inputs and states of the software, the number of tests that must be carried out for the exhaustive testing is often very large [9]. Therefore, an efficient and effective framework for the testing 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 test result.

The objective of this study is to develop a simulation-based test-bed for the white-box testing of the NPP safety-critical software. The test-bed is developed by emulating the microprocessor architecture of the safety-critical programmable logic controller (PLC) used in NPP digital I&C system and capturing its behavior at each machine instruction line. The effectiveness of the proposed software testing framework is demonstrated with the trip logic software of a fully digitalized reactor protection system (IDiPS-RPS), developed under the Korea Nuclear Instrumentation & Control Systems (KNICS) project [12].

METHODS

In this section, the test-bed development process is described. The microprocessor architecture and operation mechanism of the safety grade PLC where the safety-critical software is implemented are emulated in the test-bed.

Target System

The IDiPS-RPS is a digitalized reactor protection system developed in the KNICS project for newly constructed NPPs, as well as for upgrading existing analog based RPS [12]. It has the same function as an analog-based RPS to automatically generate a reactor trip signal and engineered safety features actuation signals whenever demand comes. IDiPS-RPS is composed of four redundant channels where each channel consists of various processors such as bistable processors (BPs), coincidence processors (CPs) [13]. 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 and CP generates a hardware-actuating trip signal by voting logic.

For the IDiPS-RPS application, the safety-graded PLC platform (POSAFE-Q) is used [14]. The POSAFE-Q consists of various modules such as a processor module, communication module, and I/O module. The processor module consists of a TI C32 DSP CPU and various types of memories such as flash memory and SRAM. The application programs of the IDiPS-RPS such as BP trip logic and CP voting logic is downloaded in the memory embedded within the processor module. The application software is developed using the function block diagram and ladder diagram (FBD/LD) programming. In the implementation, the FBD/LD programs are compiled to machine instruction which are loaded to PLC memory area and executed by the PLC microprocessor [15]. An overview of the IDiPS-RPS software structure is shown in Fig. 1.

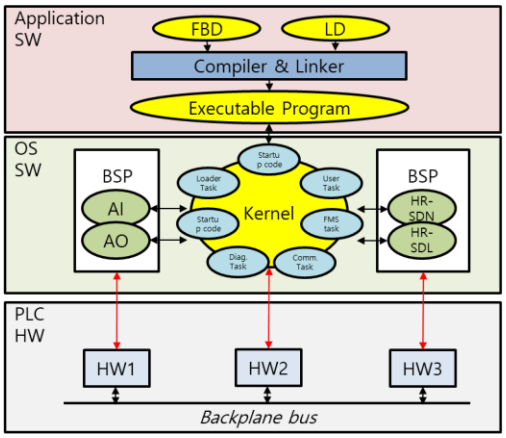

Fig. 1. The structure of IDiPS-RPS safety-critical software.

Development of the Safety-critical Software Test-bed

The fundamental characteristic of PLC operation is their cyclic operation mode. Especially, each iteration of the cyclic operation of PLC, called a scan cycle, consists of several operation stages that are sequentially repeated. After checking its own status, the equipment copies all the physical input values into its memory. Then the output of the software is updated based on the embedded software logics. These operations are repeated at a fixed interval of time, called a scan time.

In this study, a simulation-based test-bed was developed which simulates the behavior of the safety-critical software for various states of input/internal variable and validate the output of the software. The test-bed was developed in C environment where the PLC microprocessor architecture including the CPU register, memory of a POSAFE-Q is emulated and the execution characteristics (e.g. CPU register, memory access sequence) of the BP trip logic is captured at each machine instruction line. It is notable that the proposed test-bed framework can be applied to other safety grade PLC and safety-critical software for software testing.

The main part of the test-bed is the *Architecture module* which emulates the components of the 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 [16]. The second module of the test-bed is the *Assembler module* where the syntax of each PLC microprocessor instruction set which contains its specific opcode, the addressing mode, and operands are modeled. The third module of the test-bed is the *Emulation module* where the operands of the instruction set from the register files are read and its specific operation are performed. The *Interface module* provides an interface between the modules. For example, the decoded instruction set from the *Assembler module* to the *Emulation module* is transferred to conduct its operation and the operation result from the *Emulation module* is updated to the CPU contexts emulated in the *Architecture module*. Fig. 2 shows the structure of the developed test-bed.

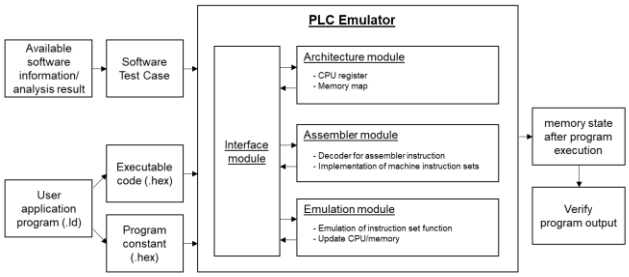

Fig. 2. An overview of the developed simulation-based test-bed for safety-critical software testing.

CASE STUDY

As a case study, the proposed software testing framework using the developed simulation-based test-bed was applied to an example safety-critical software program of IDiPS-RPS. The test procedure and results for the target software are also described.

Target Safety-critical Software

In the IDiPS-RPS, the function of each processor is implemented as a software logic in the PLC memory map. For example, 19 trip logic modules are defined in the BP software module and the process variable of each module is compared against its predefined threshold values to generate the trip signal [17]. Among 19 trip logics, the pressurizer-low pressure (PZR_PR_LO) 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 PZR_PR_LO trip logic is one of the most complicated logics among BP trip logics which

includes various functions such as operator bypass, reset delay timer and the set-point reset by operator. Fig. 3 shows the operation logic of the PZR_LO_PR trip.

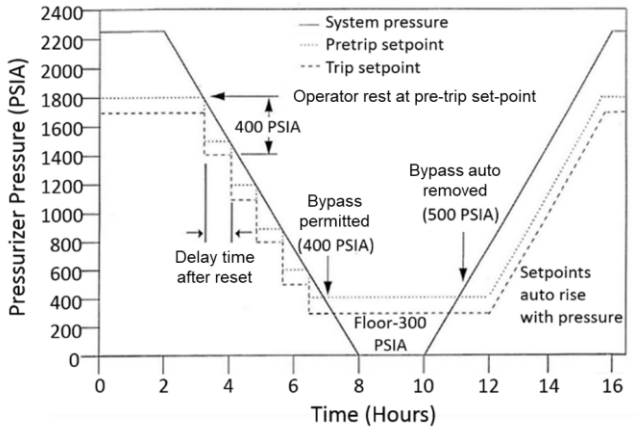

Fig. 3 An overview of the BP PZR_PR_LO trip logic [17].

Test Procedure of Target Software

In this study, the software test cases for PZR_PR_LO trip logic was developed by deriving the possible states of software input and internal variables. The test inputs for the NPP safety-critical applications are the inputs which cause the activation of protective action such as a reactor trip signal generation. Since a digital I&C system in NPP reads the inputs from instrumentation sensors as discrete digital values using an ADC, the profile of software input variable that generates the trip signal were developed in consideration of the characteristics of digital components as well as the plant thermo-hydraulic properties. For example, the profile of the process variable which is the pressurizer pressure for PZR_PR_LO trip logic is obtained from plant thermo-hydraulic simulation. As a representative pressure transient accident, a large loss of coolant accident is selected for the trip demand condition. The plant model of APR-1400 was used for estimating the plant responses using the MARS code [18], which was developed in KAERI for a thermo-hydraulic analysis of a NPP. For software internal variable, its profile was developed based on the possible range of each internal variable that generates the trip signal by inspecting the software internal logic.

Based on the obtained profile of each input and internal variable, the test cases were generated as a combination of possible states of each variable that generate the trip signal output by the software as true. In result, a total of 705,892,684 test cases were derived and 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. Fig. 4 illustrates the procedure of software testing using the simulation-based software test-bed with the test cases.

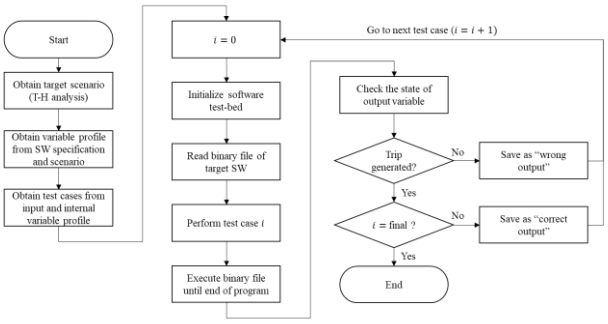

Fig. 4. Software testing procedure using developed software test-bed with test cases.

Test Result of Target 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. 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 of the PZR_PR_LO trip logic software. 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. 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. All the 705,892,684 test cases derived in the case study generated the trip signal for PZR_PR_LO trip logic and the test was conducted in 76.04 hours using sixteen 3.60 GHz logical processors (6.205 ms per test case, in average).

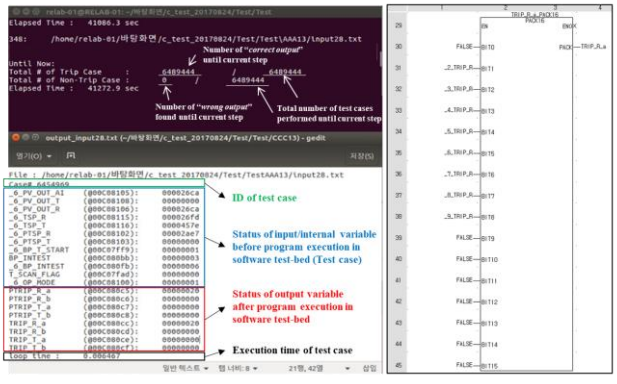

Fig. 5. An example of the test result for BP PZR_PR_LO trip logic software.

CONCLUSION

In this study, the software test method utilizing simulation-based software test-bed was proposed. The test-bed for software white-box testing was developed considering the characteristics of machine language used by the safety-critical PLC and the CPU architecture of the PLC microprocessor. As an application of the proposed software test method, a IDiPS-RPS BP software logic used in the trip signal generation of was tested based on the machine code and the test cases developed for the BP trip logic software. An important characteristic of the proposed software test approach is that the test sets can be quantitatively derived to achieve exhaustive testing of the safety-critical software. In addition, 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 black-box testing which uses trajectory inputs for software testing. 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 demand comes, other causes of software errors should be investigated in consideration of its running environment. For example, 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. Furthermore, 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 to completely model the software failure.

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. P. V. VARDE, J. G. Choi, D. Y. Lee, J. B. Han, "Reliability Analysis of Protection System of Advanced Pressurized Water Reactor-APR 1400," KAERI/TR-2468/2003, Korea Atomic Energy Research Institute (2003).
2. H. G. KANG, T. SUNG, "An analysis of safety-critical digital systems for risk-informed design," *Reliability Engineering & System Safety*, **78**, 307-314 (2002).
3. 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**, 145-148 (2007).
4. 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**, 32-43 (2007).
5. 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**, 38-49 (2013).
6. T. L. CHU, "Development of quantitative software reliability models for digital protection systems of nuclear power plants," NUREG/CR-7044, U.S. Nuclear Regulatory Commission (2013).
7. 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**, 121-132 (2007).
8. 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**, 1542-1546 (2009).
9. 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," *NPIC & HMIT 2015*, Charlotte, NC, February 22-26, 2015, American Nuclear Society (2015).
10. T. L. CHU, M. YUE, M. MARTINEZ-GURIDI, J. LEHNER, "Review of quantitative software reliability methods," BNL-94047-2010, Brookhaven National Laboratory (2010).
11. N. WALL, A. KOSSILOV, "Verification and validation of software related to nuclear power plant control and instrumentation," IAEA-12-SP-384.37, IAEA (1994).
12. K. C. KWON, M. S. LEE, "Technical review on the localized digital instrumentation and control systems," *Nuclear Engineering and Technology*, **41**, 447-454 (2009).
13. 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**, 421-428 (2012).
14. M. LEE, S. SONG, D. YUN, Development and Application of POSAFE-Q PLC Platform, IAEA-CN-194, IAEA (2012).
15. 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, May 25-26, 2006, Korean Nuclear Society (2006).
16. TEXAS INSTRUMENTS, *TMS320C3x User's guide* (1997).
17. J. G. CHOI, D. Y. LEE, "Development of RPS trip logic based on PLD technology," *Nuclear Engineering and Technology*, **44**, 697-708 (2012).
18. 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**, 1611-1642 (1999).