Contents lists available at ScienceDirect

Nuclear Engineering and Technology

journal homepage: www.elsevier.com/locate/net

Original Article

# Experimental approach to evaluate software reliability in hardware-software integrated environment

Jeongil Seo [a], Hyun Gook Kang [b], Eun-Chan Lee [c], Seung Jun Lee [a, *]

[a] Ulsan National Institute of Science and Technology, South Korea
[b] Rensselaer Polytechnic Institute, USA
[c] Korea Hydro & Nuclear Power Co., Ltd, South Korea

ARTICLE INFO

*Article history:*
Received 21 July 2019
Received in revised form
26 December 2019
Accepted 8 January 2020
Available online 17 January 2020

*Index Terms:*
Software reliability
Failure modes and effects analysis
Digital instrumentation and control
Probabilistic safety assessment

ABSTRACT

Reliability in safety-critical systems and equipment is of vital importance, so the probabilistic safety assessment (PSA) has been widely used for many years in the nuclear industry to address reliability in a quantitative manner. As many nuclear power plants (NPPs) become digitalized, evaluating the reliability of safety-critical software has become an emerging issue. Due to a lack of available methods, in many conventional PSA models only hardware reliability is addressed with the assumption that software reliability is perfect or very high compared to hardware reliability. This study focused on developing a new method of safety-critical software reliability quantification, derived from hardware-software integrated environment testing. Since the complexity of hardware and software interaction makes the possible number of test cases for exhaustive testing well beyond a practically achievable range, an importance-oriented testing method that assures the most efficient test coverage was developed. Application to the test of an actual NPP reactor protection system demonstrated the applicability of the developed method and provided insight into complex software-based system reliability.

## 1. Introduction

For the last couple of decades, digital instrumentation and control (I&C) systems in nuclear power plants (NPPs) have been replacing analog I&C systems due to the enhanced digital features as well as the scarcity of parts for existing systems and high maintenance costs. Unlike traditional analog systems, one important characteristic of digital systems is software. Software failures, differing from random hardware failures, are driven by specific inputs that lead to the failures, so quantitative software reliability assessments require a different approach than hardware reliability assessments.

Reliability in safety-critical systems and equipment is of vital importance, so the probabilistic safety assessment (PSA) has been widely used for many years in the nuclear industry to address it in a quantitative manner [1–4]. However, due to the lack of available methods, many conventional PSA models still do not properly consider software reliability. So far, safety-critical software reliability in NPPs has been assumed to be perfect or very high compared to hardware reliability without a proper foundation or experimental data. Further, it is difficult to apply the software reliability evaluation methods used in other industries to NPP software on account of the extremely low failure probability required in the nuclear industry. Therefore, it is necessary to develop a method to evaluate the reliability of safety-critical software [5–9].

Numerous studies have been conducted to develop a method to reflect software reliability in PSA models. Kim et al. showed the limitations in applying software reliability growth models to safety-critical software due to the rare failure sets in NPP software [10]. In another work, a software reliability estimation method based on software development life cycle quality showed the potential for application in the nuclear field; however, it has limitations stemming from its reliance on expert judgement [11]. Test-based methods have been proposed using white-box and black-box testing [12–14]. Black-box testing approaches have limitations regarding the coverage and completeness of the test cases. In contrast, while white-box testing methods could reflect the operational profile of software, the number of test cases is often very large. IEC 61226 classifies the risk potential for safety functions into four safety integrity levels (SILs), with safety-critical software

* Corresponding author.
*E-mail address:* sjlee420@unist.ac.kr (S.J. Lee).

https://doi.org/10.1016/j.net.2020.01.004

failure probability per demand assumed to be SIL 4, signifying the lowest potential for failure. Specifically, this means that the probability of a dangerous failure of a safety function on demand is between 10E-4 and 10E-5, although this range has no experimental basis.

The purpose of this work is to develop a software reliability evaluation method using a hardware-software integrated testing environment. This work is a part of a larger project to quantitatively evaluate software reliability in a digital reactor protection system (RPS). In this project, two methods were proposed: simulation-based software testing for exhaustive testing [15], and evaluation of the entire environment using a hardware-software integrated testing environment. The first method seeks to prove the completeness of the application logics, while the second method, considering hardware-software integrated testing, confirms that the software application logics perform their functions reliably in real operating situations. While the emulator used in the first approach can perform tests much faster than the second approach (which uses actual hardware), it is not easy to guarantee the reliability of test results. On the other hand, exhaustive testing is not possible with the second approach. Despite this, more reliable results can be obtained because the hardware-software integrated test examines software behavior in an actual operating environment. These two approaches are therefore complementary, with both experimental methods necessary for software reliability evaluation. This research focuses on the second.

## 2. Target system

The target system of this work is the digitalized RPS of the Advanced Power Reactor 1400 (APR1400), which is a fully digitalized NPP. The RPS of the APR1400 consists of four redundant channels, each of which contains two bistable processors (BPs) and three coincidence processors (CPs) [16,17]. Fig. 1 shows a block diagram of the IDiPS-RPS, which is a prototype of the APR-1400 RPS.

The BPs receive an input signal generated by individual sensors and discrete signals from the reactor core protection system (RCOPS). The input parameter is periodically scanned by the BPs for each of the safety parameters. This digital RPS possesses fifteen types of trip parameters, including variable overpower trip, high log power trip, high log power density (LPD) trip, departure from a nucleate boiling ratio (DNBR) trip, high and low pressurizer (PZR) pressure trips, high and low steam generator (SG) A/B level trips, low SGA/B pressure trip, low SG A/B reactor coolant flow trip, and high containment pressure trip. The high LPD and low DNBR trip are determined in the RCOPS and the results are transferred to eight BPs. For the other 13 trip logics, a BP determines the trip state by comparing the measured process variables with a pre-determined trip set-point. Then a partial trip signal is transmitted from the BPs to the CPs. The CPs then generate a trip signal based on a 2-out-of-4 voting logic for each process parameter based on the signal from the BPs.

A BP transfers partial trip signals with their quality signals to four channels of CPs. The quality signal indicates the reliability of the generated partial trip signal in the BP. When any hardware in a BP fails and gets detected by diagnostic functions, a bad quality signal is transferred to the connected CPs, indicating to not use the partial trip signal with bad quality in the voting logic for final trip signal generation. Partial trip signals consist of RPS and RCOPS partial trip signals, each with its own quality signal.

## 3. Methodology

The objective here is to conduct software reliability testing considering hardware, software, and the operating system to check whether the software works properly and provide quantitative software reliability data.

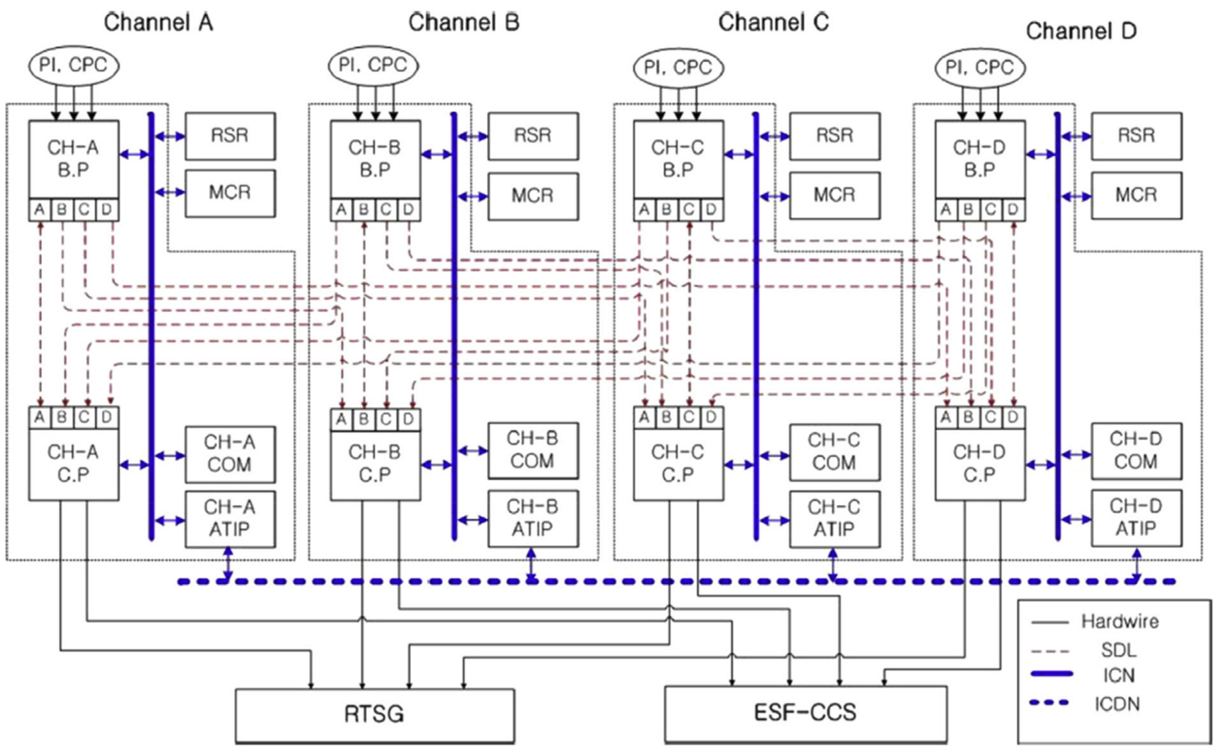

**Fig. 1.** Block diagram of the IDiPS-RPS [16].

Since the total number of input cases is huge and the RPS requires extremely high reliability, a simple input coverage which is the number of examined test cases divided by the total number of test cases cannot be used. For example, if there are 10,000 test cases, 9999 tests should be performed to make 99.99% coverage. Due to the extremely high coverage required, this approach is almost same as an exhaustive testing. Therefore, a coverage considering the importance of each input case is proposed.

Generally, it is not practical to predict the occurrence probability or frequency of a certain input set. As the plant parameters and the controls by operators in a BP input set and the partial trip signals in a CP input set are determined by abnormal situation occurrence, it is not practical to evaluate weightings for certain test cases unless the frequencies of the abnormal situations and operator actions can first be quantitatively estimated. On the other hand, the quality signals in a test case would have different occurrence probabilities because diagnosis results are included. Note here that the transferred signals from a BP to a CP consist of partial trip signals and their qualities—since a quality signal indicates whether the corresponding component is normal or not, the occurrence possibility of a quality signal can be predicted based on the related hardware failure rate. This relative occurrence possibility of a test set is used to evaluate the importance in this work. For example, the importance of a test set is high when it includes one ‘bad’ quality signal related to a hardware failure mode with a relatively higher failure rate. In contrast, a test case representing multiple simultaneous hardware failures has relatively low importance.

While both BP and CP test cases include quality signals representing hardware failures, the importance method proposed in this work is applied to only CP test cases because the total number of BP test cases is an executable number for exhaustive testing. Therefore, the methods for developing test cases in this work are categorized into two: exhaustive test case generation for BP testing, in which the test cases have even importance, and selective test case generation to meet the target coverage for CP testing, in which the test cases have different importance designations.

In the proposed method, the coverage of CP testing is defined as the ratio of the importance of performed test cases to that of completed test cases, as shown in Equation (1). The denominator in Equation (1) indicates the total importance of all possible test cases and the numerator represents the importance of tested test cases.

$$coverage = \frac{\sum_{i=1}^{t} IP_i}{IP_{total}} \tag{1}$$

where,

$IP_i$: Importance of $i$th test case
$IP_{total}$: Total importance of complete test cases
t: Number of tested test cases

Since a CP receives inputs from eight BPs, the CP test cases consist of the failure mode combinations of the eight BPs. After figuring out possible failure mode combinations from one BP, the denominator of the coverage of total CP test cases can be derived. Since each test case includes at least one failure, the denominator can be calculated by Equation (2) below.

$$IP_{total} = 1 - \prod_{i=1}^{8} \prod_{j=1}^{nf} \left(1 - p_{ij}\right) \tag{2}$$

where,

$IP_{total}$: Total importance of all CP test cases
$nf$: Number of failure modes in a BP
$p_{ij}$: Unavailability by failure mode $j$ in $i$th BP

$p_{ij}$ represents the component unavailability by a failure mode. According to NUREG-0492, for periodic tests performed at intervals of $T$, the unavailability, $q(t)$, rises from a low of $q(t = 0) = 0$ immediately after a test is performed to a high value of immediately before the next test is performed. Since the exponential can be approximated by a linear function (for $\lambda T < 0.1$) and the failure rate of RPS hardware is very small, the average unavailability between tests is approximately $\lambda T/2$[6]. It is assumed that all components in the target system have the same periodic test intervals, because the importance proposed in this work is a relative measurement to compare the occurrence possibilities of test cases.

The numerator of the coverage can be found by combining the importance of the test cases performed. The importance of each test case is calculated by Equation (3). If $k$th CP test case includes one hardware failure, $IP_k$ is the product of the unavailability of the failed hardware and the availabilities of the other components.

$$IP_k = \prod_{i=1}^{8} \prod_{j=1}^{nf} x_{ij} \tag{3}$$

where,

$IP_k$: Importance of $k$th CP test case
$nf$: Number of failure modes in a BP
$x_{ij}$: If the $j$th failure mode in the $i$th BP is true, then $x_{ij} = p_{ij}$, otherwise $x_{ij} = (1 - p_{ij})$
$p_{ij}$: Unavailability by failure mode $j$ in $i$th BP

To perform such quantitative testing, the following procedure is suggested: (i) set the experimental environment as close as possible to an actual operating environment; (ii) identify which hardware input signal sets match the software application logic inputs; (iii) generate test cases according to every possible input; (iv) optimize the number of test cases to achieve a sufficiently high reliability; and (v) execute the test cases. There is no need to perform test cases below a certain level when performing hardware-software integrated testing; such particular levels can be determined through measurements of importance using failure rate data from the failure modes and effects analysis (FMEA) of software-related hardware. FMEA proposes possible failure modes for each hardware component and analyzes the probability of each failure mode and its final effect [11].

To assess the software reliability method proposed in this work, this research quantitatively evaluates software reliability in the APR1400 RPS by including importance measurements and test case optimization. By prioritizing the test cases in terms of importance and executing only the high-importance ones, a sufficiently high reliability can be ensured in a relatively short period of time.

## 4. Test environment and test cases development

### 4.1. Hardware-software integrated test environment

As mentioned above, the APR1400 RPS consists of eight BPs and twelve CPs. Instead of using all eight BPs and twelve CPs, only one BP and one CP, which are the same as the actual hardware installed in the APR1400 RPS, are used in this study to construct the experimental environment. In order to replace the seven BPs and eleven CPs not included in this experimental environment, an input/output (I/O) simulator is used to generate virtual signals and simulate plant parameters such as temperature and pressure. The I/O simulator generates and delivers the plant parameters necessary for BP application execution. Although the experimental environment uses only two real hardware modules because of their cost, as shown in Fig. 2, it can simulate the same safety-related signals as a real hardware-software integrated environment. The experimental environment has a total of 91 signals that are transferred to the actual hardware BP and CP. The integrated testing carried out in this study is aimed at an overall inspection of the safety-related functions in a fully integrated state, specifically focusing on the reactor trip signal.

Among the eight BP signals entering the CP-D1, the BP-D1 signal is generated by an actual processor and input to the CP via safety data link (SDL). The remaining seven virtual signals, labeled BP-A1, BP-A2, BP-B1, BP-B2, BP-C1, BP-C2, and BP-D2, are generated in the I/O simulator and input to the CP-D1 through the SDL. All necessary plant parameters are generated in the I/O simulator and delivered to the analog or digital input module of the BP.

Fig. 3 depicts the development process for the experimental environment using the I/O simulator application. The application was developed using pSET2 so that when a developer enters a test case into the application, the I/O simulator can act as an input for the corresponding set of signals into the relevant hardware [18]. The pSET2 is an engineering tool that can translate function block diagram (FBD) programs into executable codes for programmable logic controllers (PLCs).

## 4.2. Composition of each test case

The effects of failures in an RPS are classified into three categories, as shown in Fig. 4. If there is no end effect in a given failure mode, the hardware status is normal. If an end effect that is not associated with a safety function occurs, such as a light fixture failure, it is classified as a failure mode without an end effect. When a failure mode does result in an end effect, it can be classified depending on whether the failure is detected or not. In the case of an undetected failure ((a) in Fig. 4), the situation can be conservatively modeled by converting the partial trip signal of the relevant test case from ‘trip’ to ‘non-trip’. That means a failure causes wrong system behavior without detection. In the case of a detected failure ((b) in Fig. 4), the situation can be simulated by setting the relevant partial quality signal to ‘bad’. This research considers all (a), (b), and normal hardware states as test cases.

All test cases were developed with the following assumptions:

1) Regarding plant parameters entered to the BP, among the huge number of combinations of plant parameters, only 15 combinations are considered. Among the 15 combinations, 13 combinations are representative of 13 trip parameter trip logics and 2 combinations are representative of 2 RCOPS trip logics.
2) For conservatism, only trip logic is considered in a test case. The cases that two or more trip logics occur simultaneously are not considered. For example, it is assumed that low DNBR is the first reactor trip variable when a loss of coolant accident or steam generator tube rupture occurs; if this first parameter does not generate a reactor trip signal, it is assumed that the function of the RPS fails.

### 4.2.1. BP test cases

The signals that are input to the BP-D1 using the I/O simulator consist of 49 signals. Fig. 5 shows the connections between the I/O simulator and BP-D1. Among the 49 signals, 24 are generated by the ex-core neutron flux monitoring system (ENFMS), auxiliary process cabinet system (APC-S), and RCOPS such as PZR pressure, SG water level and pressure, and containment pressure as shown in Fig. 2. These 24 signals are set to predefined variables to generate a specific reactor trip signal among 15 trip logics. Followings are the 15 trip logics employed in the BP application:

- Variable overpower trip
- High log power trip
- High PZR pressure trip
- Low PZR pressure trip
- Low SG 1 water level trip
- Low SG 2 water level trip
- High SG 1 water level trip
- High SG 2 water level trip
- Low SG 1 pressure trip
- Low SG 2 pressure trip
- High containment pressure trip

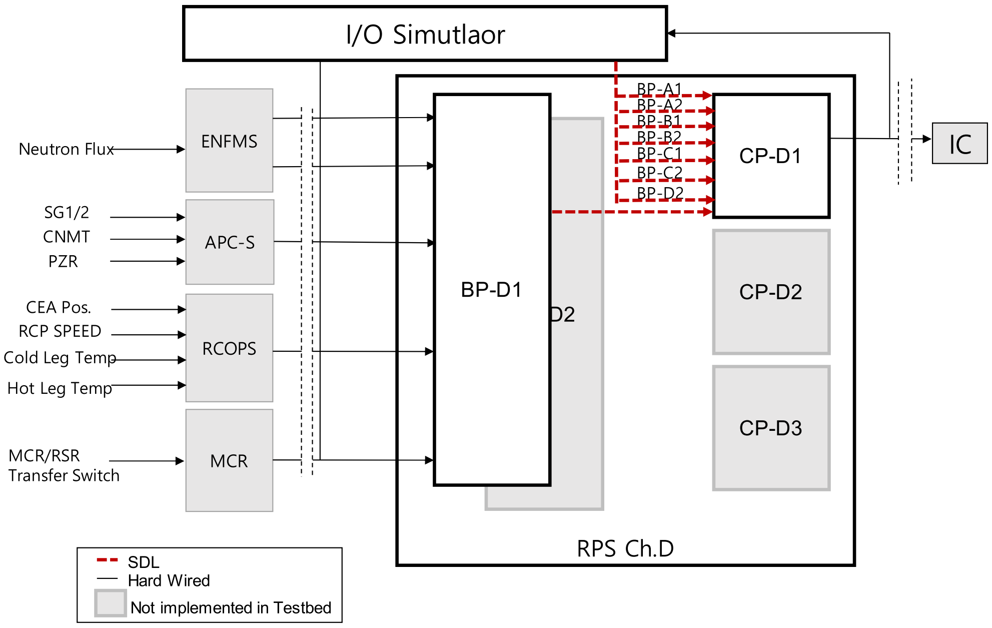

**Fig. 2.** Diagram of the experimental environment.

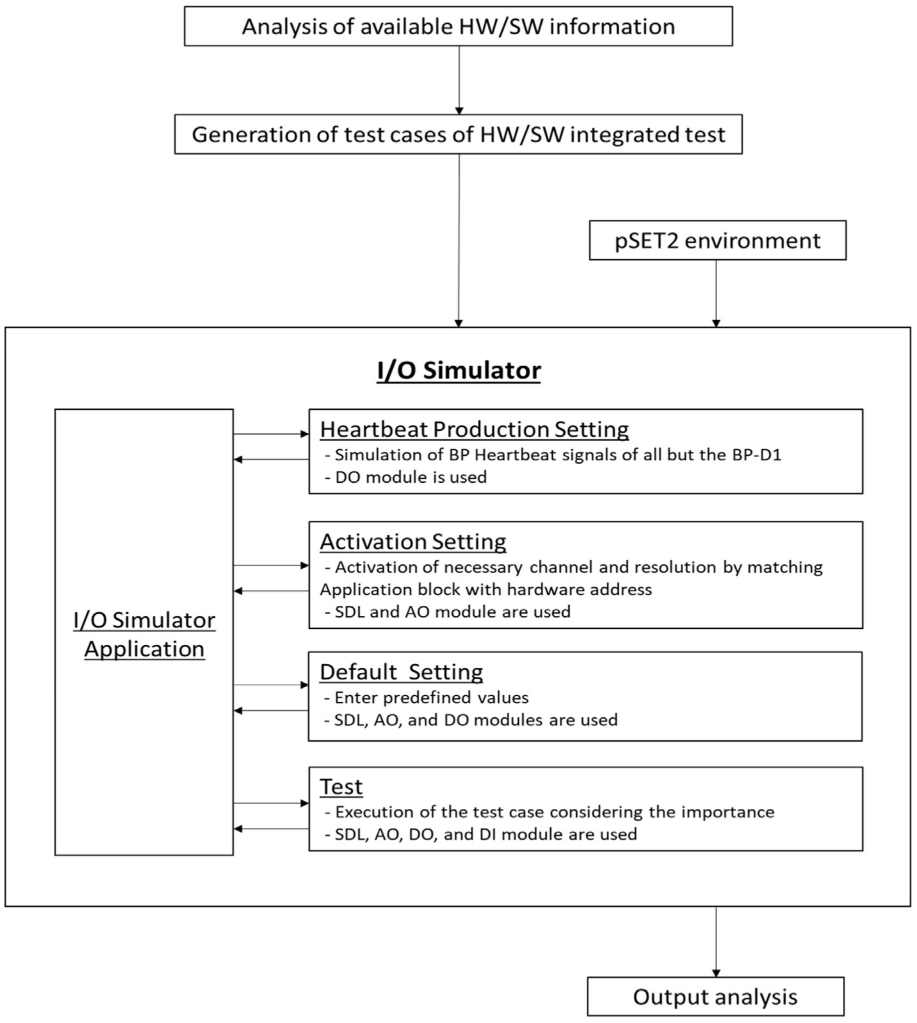
Analysis of available HW/SW information
Generation of test cases of HW/SW integrated test
pSET2 environment
I/O Simulator
I/O Simulator Application
Heartbeat Production Setting
- Simulation of BP Heartbeat signals of all but the BP-D1
- DO module is used
Activation Setting
- Activation of necessary channel and resolution by matching Application block with hardware address
- SDL and AO module are used
Default Setting
- Enter predefined values
- SDL, AO, and DO modules are used
Test
- Execution of the test case considering the importance
- SDL, AO, DO, and DI module are used
Output analysis

**Fig. 3.** Development process of the experiment environment.

- Low SG 1 RC flow trip
- Low SG 2 RC flow trip
- Low DNBR trip
- High LPD trip

For example, PZR pressure is set to 0 and the others are set to normal value at full-power operation for low PZR pressure trip. To trigger low SG 1 water level trip, the narrow water level range of the SG 1 is set to 0 and the others are set to normal.

The other 25 signals are control signals made by operators including the signals from main control room/remote shutdown room (MCR/RSR) and MCR/RCR transfer switch status signal. The 25 variables can be classified as follows:

1 MCR/RSR transfer switch status signal: 1
2 Signals from MCR: 16
3 Signals from RSR: 8

The RSR is not used in normal NPP operating environments, as it is only employed in emergency cases when the MCR is disabled. The MCR and RSR signals were considered independently according to the MCR/RSR transfer switch status.

Signals from the MCR and RSR consist of signals by operators and their quality. The signals generated by operators aim to reset a set point or bypass the certain trip logic to prevent unnecessary reactor shutdown. Four signals from the MCR are variable set point (VSP) reset signals and their quality signals, 6 of them are operating bypass (OPB) request signals and their quality signals, and 6 of them are OPB off signals and their qualities as follows:

- PZR pressure VSP reset and quality
- SG pressure VSP reset and quality
- High log power level OPB request and quality
- High log power level OPB request off and quality
- High LPD and low DNBR OPB request and quality
- High LPD and low DNBR OPB request off and quality
- Low PZR pressure OPB request and quality
- Low PZR pressure OPB request off and quality

All signals from the MCR are Boolean and are transferred from the I/O simulator to the BP-D1 via SDL. The number of BP MCR test cases is 65,536 following the combination of 16 Boolean variables.

The RSR signals are simpler than MCR signals. Only two VSP reset signals and one OBP signal are included.

- PZR pressure VSP reset and quality
- SG pressure VSP reset and quality
- Low PZR pressure OPB request and quality

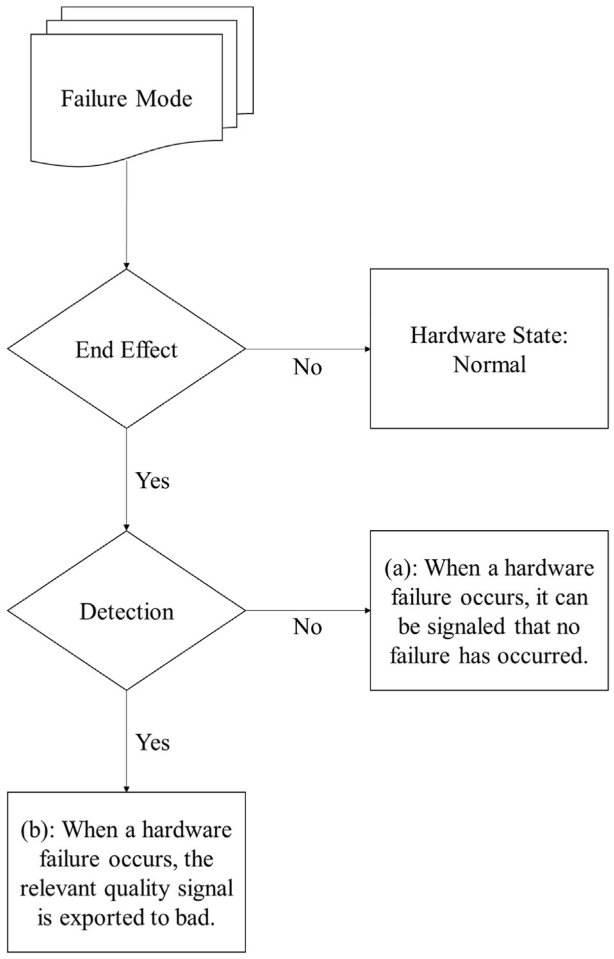
Failure Mode
End Effect
No
Hardware State: Normal
Yes
Detection
No
(a): When a hardware failure occurs, it can be signaled that no failure has occurred.
Yes
(b): When a hardware failure occurs, the relevant quality signal is exported to bad.

**Fig. 4.** Flowchart for classifying failure modes.

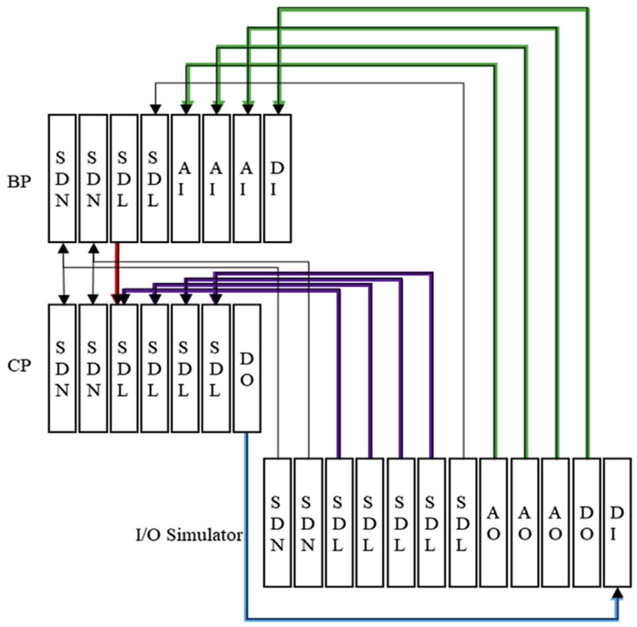

**Fig. 5.** Signals from the I/O simulator to BP-D1 and CP-D1.

- Low PZR pressure OPB request off and quality

The signals transmitted from the RSR to the BP are Boolean and transferred from the I/O simulator to the BP-D1 via SDL. The number of BP RSR test cases is 256 following the combination of 8 Boolean variables.

The total number of BP test cases is therefore 986,880 by Equation (4) below, considering MCR and RSR test cases and 15 trip logics.

$$n_{bp} = n_{trip} \times \left(n_{bp_M} + n_{bp_R}\right) \tag{4}$$

where,

$n_{bp}$: Total number of BP test cases.
$n_{trip}$: Number of trip logics.
$n_{bp_M}$: Number of BP MCR test cases.
$n_{bp_R}$: Number of BP RSR test cases.

### 4.2.2. *CP test cases*

The CP test cases consist of the signals from the BP-D1 (actual hardware) and other virtual BPs. The desired BP-D1 outputs are generated by controlling the BP-D1 input signals, while the outputs of the other BPs are directly generated by the I/O simulator.

One CP test case includes 91 variables. Under the assumption mentioned in the previous section, 49 variables transmitted from the I/O simulator to the BP-D1 are predefined to generate a specific partial trip signal. The other 42 variables are transmitted from the I/O simulator to the CP-D1, and all of them are 32-bit Boolean variables. Fourteen variables are to generate partial trip signals in seven virtual BPs. Thus, 14, 7, and 7 variables represent the quality signals for the partial trip signals, the heartbeat (HB) signals, and miscellaneous signals for virtual BPs, respectively, as follows:

1 Trip parameter partial trip quality: These quality signals cover the 13 trip logics determined by trip parameters. Each is a 32-bit Boolean variable with only 18 out of the 32 bits used. The other 14 bits have no meaning and are set to false.
2 RCOPS partial trip quality: These quality signals cover the 2 trip logics determined by the RCOPS. Each is a 32-bit Boolean variable with only 2 out of the 32 bits used. The other 30 bits have no meaning and set to false.
3 Miscellaneous: These signals include PM (Processor Module) quality and function enable quality; 2 out of 32 bits have meaning.
4 BP HB: This signal represents the HB transmitted from the connected BP. Only two cases are considered: stuck and normal.

The number of exhaustive CP signal combinations is 3.68E56 by Equation (5) below.

$$n_{cp} = n_{trip} \times \prod_{i=1}^{8} \left(n_{i,RPS} \times n_{i,RCOPS} \times n_{i,MISC} \times n_{i,HB}\right) \tag{5}$$

where,

$n_{cp}$: Total number of CP test cases
$n_{trip}$: Number of trip logics
$n_{i,RPS}$: Number of combinations of trip parameter quality signals for $i$th BP
$n_{i,RCOPS}$: Number of combinations of RCOPS quality signals for $i$th BP
$n_{i,MISC}$: Number of combinations of MISC signals for $i$th BP
$n_{i,HB}$: Number of HB inputs for $i$th BP

**Table 1**
Source of Trip parameter/RCOPS Quality Signals.

| Signal | Source |
|---|---|
| Trip parameter partial trip quality signal by certain trip logic A | PM error, AI slot_w module error, AI slot_w channel_a error |
| Trip parameter partial trip quality signal by certain trip logic B | PM error, AI slot_x module error, AI slot_x channel_a error |
| … | … |
| RCOPS partial trip quality signal by certain trip logic X | PM error, DI slot_z module error, DI slot_z channel_b error |
| RCOPS partial trip quality signal by certain trip logic Y | PM error, DI slot_z module error, DI slot_z channel_d error |

## 5. Importance of test cases

While the total number of BP test cases, 98,830, is an executable number by exhaustive testing, that of CP test cases is too huge to be exhaustively tested. Therefore, in this work, the importance of each CP test cases was analyzed.

### 5.1. Optimization of the test cases

The first step is to identify the impossible test cases. In a CP test case, there are 23 bits related to hardware failures per one BP. As mentioned previously, while partial trip signals and operator signals do not have weightings, quality signals representing hardware failures have different occurrence probabilities. In an input set from a BP to a CP, there are 23 bits related to hardware failures: 18 bits in trip parameter partial trip quality, 2 bits in RCOPS partial trip quality, 2 bits in miscellaneous, and 1 bit in HB.

To optimize the test cases by eliminating impossible combinations of these bits, it is necessary to check which hardware failures each signal is affected by, as shown in Table 1.

Impossible test cases can be identified by analyzing how the hardware failure affects the quality signal, as shown in Table 1. For example, if a PM in a BP fails, all quality signals are set to 'bad.' Also, the function enable quality signal is set to 'bad' and HB is stuck. For other hardware failures such as AI (Analog Input) module failure or channel failure, the quality signals related to the failed hardware are set to 'bad'.

Table 2 shows examples of the relationship between hardware failure(s) and the number of affected quality signals.

After analyzing the relations between hardware failures and affected quality signals, it was observed that only 293,762 possible cases of 32 quality signals are valid in a BP. Then, the number of total CP test cases is calculated as 8.32E44 by Equation (5).

### 5.2. Importance measurements

The next step is to evaluate the importance of each test case. Since failure rates are used to evaluate importance, all the failure modes are analyzed based on FMEA. Examples of hardware failure rates derived from FMEA are shown in Table 3.

Table 4 shows examples of test case importance derived from hardware failure rates by Equation (3) with an assumption that the periodic test interval is one day. There is no effect of the assumed periodic test interval on the result because all components have the same periodic test interval and the importance is relative measure. For example, failure mode 1 represents a single failure of AI module #1 in BP-A1. The importance of this failure mode is calculated by the unavailability of the failed AI module and the availabilities of the other failure modes. In total, 24 failure modes are included in a single failure of the AI module, because there are three AI modules in eight BPs. The importance of test cases with one failed AI module is 2.14E-3 because all AI modules have the same failure rate. If a failure mode includes two DI (Digital Input) module channel failures and one AI module channel failure, as shown in failure mode 6 in Table 4, then the importance of this test case is calculated as 3.99E-13.

Each BP has 293,762 possible failure modes. The total importance of all CP test cases is calculated as 0.9843 by Equation (2). Then, the coverage is determined by the number of test cases examined.

## 6. Result analysis

A CP testing coverage of 1 can be satisfied by performing the entire set of test cases, a total of 8.32E44, but this would take an extremely long time. To ensure a sufficiently high degree of reliability in the hardware-software integrated environment, the importance of the test cases was measured, and in order to ensure maximum reliability through minimum test case execution, the test cases should be performed in order of importance. Coverage when the *n*th test case is performed can be derived as Equation (1). Since there is a large difference in the importance value of each test case, the result is expected to increase exponentially as the value of *n* increases. High reliability can be ensured more efficiently by setting a coverage target.

In the experiment performed in this work, the test cases were

**Table 3**
Examples of hardware failure rates [19].

| Hardware failure mode | Failure rate [/24 h] |
|---|---|
| AI module error | 2.15E-03 |
| DI module error | 2.83E-04 |
| PM error | 1.36E-04 |
| DI channel error | 7.89E-05 |
| AI channel error | 6.41E-07 |

**Table 2**
Examples of the number of quality bit signals affected by failure Mode(s).

| Hardware failure(s) | Number of affected quality bit signals by the hardware failure(s) |
|---|---|
| Channel a error in AI module X | 2 |
| AI module X error | 9 |
| Channel b error in DI module Y | 1 |
| DI module Y error | 4 |
| PM Z error | 23 |
| PM module Z error + channel a error in AI module X | 23 |
| PM module Z error + AI module X error | 23 |

**Table 4**
Examples of CP test case importance.

| No | Failure mode | Importance of test case |
|---|---|---|
| 1 | BP-A1 AI module#1 error | 2.14E-03 |
| 2 | BP-A1 DI module#1 error | 2.83E-04 |
| 3 | BP-A1 DI module#1 error + BP-B1 DI module#1 error | 8.00E-08 |
| 4 | BP-A1 AI module#1 error + BP-B1 DI module#1 error + BP-B1 AI module#1 error | 1.31E-09 |
| 5 | BP-A1 module#1 error + BP-B1 DI module#1 error + BP-C1 DI module#1 channel#3 error | 4.81E-11 |
| 6 | BP-A1 DI module#1 channel#2 error + BP-B1 DI module#1 channel#1 error + BP-C1 AI module#1 channel#2 error | 3.99E-13 |

optimized with a target coverage of 0.9999 according to the SIL 4 requirement. As shown in Fig. 6 and Table 5, a coverage of 0.99991136 can be satisfied by performing only 672,000 CP test cases. Since the test cases with the highest importance are tested first, it is confirmed that after an initial rapid increase in coverage, further testing only yields minimal, gradual increases in coverage.

Through this study, 1,658,880 test cases were performed in the hardware-software integrated environment, and no failures were found. Since each test takes 120 ms in the implemented environment, it took 32.9 h to perform 986,880 BP test cases and 22.4 h for 672,000 CP test cases to achieve 0.9999 coverage.

The purpose of this work is to derive the software reliability which can be used as the basic event in a PSA model. Equation (6) represents the failure probability of the RPS software. $p_i$ indicates the relative occurrence probability of a test case and the sum of all $p_i$ should be 1.

$$\theta = \sum_{i=1}^{n} \theta_i p_i \tag{6}$$

Where,

$\theta$: the failure probability of the RPS software

$\theta_i$: the failure probability of the RPS software for $\underline{i}$th test case. If the test result is failure, then $\theta_i = 1$, else $\theta_i = 0$

$p_i$: the relative occurrence probability of the $i$th test case

Since both the importance and $p_i$ represent the relative occurrence probability of a test case and both sum of all importance and all $p_i$ are 1, the importance can be used for the $p_i$ in Equation (6). $\theta_i$ for the test cases examined in the experiment are 0 because there is no failed output observed. If all test cases not examined are assumed to be failed ($\theta_i = 1$), then the coverage can be used as the reliability by the equation.

## 7. Conclusion

Quantitatively proving software reliability or providing an empirical basis is important in the nuclear industry because most current PSA models assume the reliability of software to be 1 or very high. In the nuclear field, while exhaustive testing could be one possible option to prove the extremely high reliability of safety-critical software because of the relatively simple logics and small number of inputs, the number of the exhaustive test cases is impractical to be performed. Therefore, this study proposed a method to efficiently perform software testing based on occurrence probabilities of the test cases.

By evaluating the importance of each test case and testing sequentially from those with the highest importance, a high level of software reliability can be achieved in a relatively short period of testing. In the experiment performed in this study, it was proven that 0.9999 coverage was satisfied by executing 672,000 CP test cases among the total 8.32E44 test cases.

The implemented experiment environment includes only two PLC racks of the RPS. To cope with this partially implemented experiment environment, the I/O simulator was developed to provide necessary signals to the CP. However, due to this experimental environment, some part of the RPS such as the voting logic of four redundant CP channels cannot be tested in this work. Moreover, there could be some unexpected failures if full PLCs are employed. For more accurate and reliable evaluation result, full

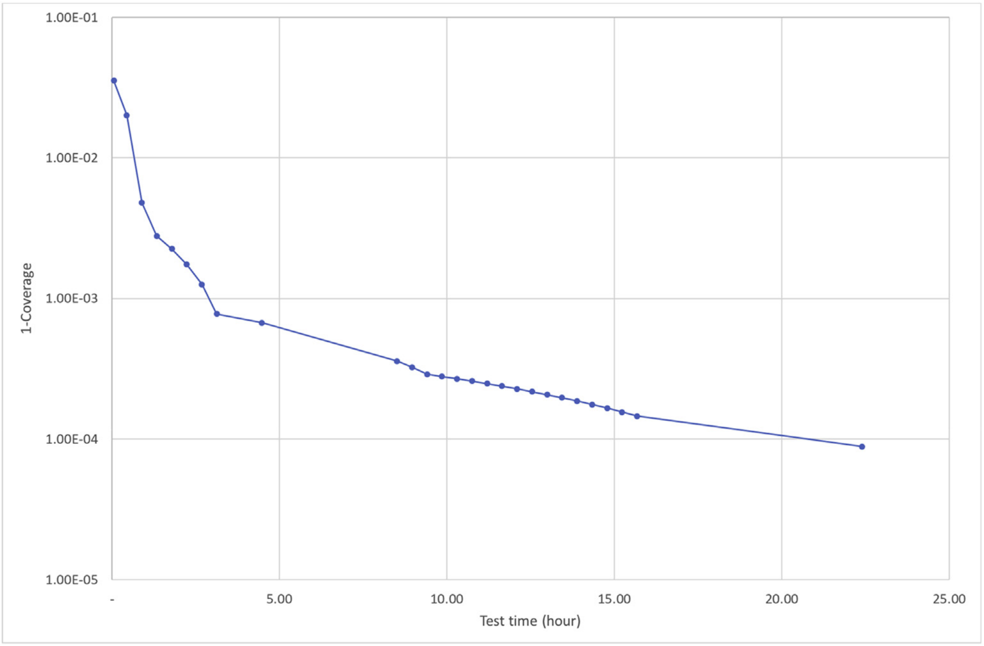

**Fig. 6.** Coverage of CP testing.

**Table 5**
Coverage according to the number of CP test cases.

| # of test cases | Test time [hour] (120 ms/test) | Coverage |
|---|---|---|
| 1920 | 0.06 | 0.96455830 |
| 13,440 | 0.45 | 0.97987931 |
| 94,080 | 3.14 | 0.99922413 |
| 672,000 | 22.40 | 0.99991136 |
| 1,424,640 | 47.48 | 0.99995783 |
| 2,392,320 | 80.00 | 0.99998739 |
| 3,682,560 | 122.75 | 0.99998953 |
| 4,448,640 | 148.28 | 0.99998985 |
| 5,295,360 | 176.51 | 0.99999017 |

scope experiment is needed to be performed.

The proposed method can provide insights to evaluators in determining target coverages efficiently by analyzing the trend of coverage increase. In terms of regulation, a standard importance value for satisfying a given target coverage can be used as a screen-out criterion for test case execution. Also, this approach could be applied to safety-critical software that is relatively simple but requires very high reliability in other safety-critical infrastructures.

## Acknowledgment

This work has supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2018M2B2B1065653).

## References

[1] T. Aldemir, D.W. Miller, M.P. Stovsky, J. Kirschenbaum, P. Bucci, A.W. Fentiman, L.T. Mangan, Current State of Reliability Modeling Methodologies for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments" United States Nuclear Regulatory Commission, vol. 6901, U.S.NRC, USA, 2004. NUREG/CR.
[2] T. Aldemir, M.P. Stovsky, J. Kirschenbaum, D. Mandelli, P. Bucci, L.A. Mangan, D.W. Miller, X. Sun, E. Ekici, S. Guarro, M. Yau, B. Johnson, C. Elks, S.A. Arndt, Dynamic Reliability Modeling of Digital Instrumentation and Control Systems for Nuclear Reactor Probabilistic Risk Assessments, U.S.NRC, USA, 2004. NUREG/CR-6942.
[3] T. Aldemir, S. Guarro, J. Kirschenbaum, D. Mandelli, L.A. Mangan, P. Bucci, M. Yau, B. Johnson, C. Elks, E. Ekici, M.P. Stovsky, D.W. Miller, X. Sun, S.A. Amdt, Q. Nguyen, J. Dion, A Benchmark Implementation of Two Dynamic Methodologies for the Reliability Modeling of Digital Instrumentation and Control Systems, vol. 6985, U.S.NRC, , USA, 2004. NUREG/CR.
[4] T.L. Chu, M. Yue, G. Martinez-Guridi, K. Memick, J. Lehner, A. Kuritzky, Modeling a Digital Feedwater Control System Using Traditional Probabilistic Risk Assessment Methods, vol. 6997, U.S.NRC, , USA, 2004. NUREG/CR.
[5] S.J. Lee, W.D. Jung, J.E. Yang, PSA model with consideration of the effect of fault-tolerant techniques in digital I&C systems", Ann. Nucl. Energy 87 (2010).
[6] S.J. Lee, J.G. Choi, H.G. Kang, S.C. Jang, Reliability assessment method for NPP digital I&C systems considering the effect of automatic periodic tests, Ann. Nucl. Energy 37 (2010).
[7] H.G. Kang, M.C. Kim, S.J. Lee, H.J. Lee, H.S. Eom, J.G. Choi, S.C. Jang, An overview of risk quantification issues of digitalized nuclear power plants using static fault tree, Nucl. Eng. Technol. 41 (2009).
[8] H.G. Kang, T. Sung, An analysis of safety-critical digital systems for risk-informed design, Reliab. Eng. Syst. Saf. 78 (2002).
[9] J.H. Jo, S.J. Lee, W.D. Jeong, Fault analysis of reactor protection system based on FMEA, 2014. KAERI, ROK, KAERI/TR-5655.
[10] M.C. Kim, S.C. Jang, J. Ha, Possibilities and limitations of applying software reliability growth models to safety critical software, Nucl. Eng. Technol. 39 (2007).
[11] 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, Ann. Nucl. Energy 51 (2013).
[12] 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.
[13] S. Kuball, J.H.R. May, A discussion of statistical testing on a safety-related application, Proc. Inst. Mech. Eng. O J. Risk Reliab. 221 (2007) 121e132.
[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, Reliab. Eng. Syst. Saf. 94 (2009) 1542e1546.
[15] S.H. Lee, S.J. Lee, J.K. Park, E.C. Lee, H.G. Kang, Development of simulation-based testing environment for safety-critical software, Nucl. Eng. Technol. 50 (2018).
[16] P.V. Varde, J.G. Choi, D.Y. Lee, J.B. Han, Reliability Analysis of Protection System of Advanced Pressurized Water Reactor - APR 1400 vol. 2468, Korea Atomic Energy Research Institute (KAERI), Republic of Korea (ROK), 2003. KAERI/TR.
[17] C.G. Lee, I.S. Oh, D.H. Kim, J.H. Park, J.H. Shin, Y.B. Kim, Requirements for the Development of KNICS Control Systems, 2004. KAERI, ROK, KAERI/TR-2737.
[18] J.B. Yoo, S.D. Cha, E.K. Jee, in: A Verification Framework for FBD Based Software in Nuclear Power Plants" 15th Asia-Pacific Software Engineering Conference, 2008.
[19] S.J. Lee, W. Jung, J. Yang, PSA Model Considering the Effects of Self-Diagnostic of Digital I&C Systems, Korea Atomic Energy Research Institute (KAERI), Republic of Korea (ROK), 2015. KAERI/TR-5946.