A Specific Method for Software Reliability of Digital Controller in NPP

Young Jun Lee, Jong Yong Keum, Jang Soo Lee
I&C/HF Division
Korea Atomic Energy Research Institute
Daejeon, Korea
e-mail: yjlee426@kaeri.re.kr, jykeum@kaeri.re.kr, jslee@kaeri.re.kr

Young Kuk Kim
Department of Computer Science & Engineering
Chungnam National University
Daejeon, Korea
e-mail: ykim@cnu.ac.kr

Abstract— **Most new controllers used in the safety systems of nuclear power plants have been developed using digital systems, and conventional analog controllers are also increasingly being replaced with digital controllers. Therefore, the importance of software that operates within the digital controller of a nuclear power plant is further increased. This paper describes a reliability evaluation method for the software to be used for a specific operation of a digital nuclear power controller. It is possible to calculate the software reliability when obtaining the failure rate and utilizing the existing calculation method. We attempt to achieve differentiation by creating a new definition of the fault, imitating the software fault using the hardware, and giving the consideration and weights for injection faults.**

Keywords- software reliability; digital controller in NPP; software life cycle; fault injection.

I. INTRODUCTION

To ensure the safety of software used in a Nuclear Power Plant (NPP), the Nuclear Regulatory Commission (NRC), the nuclear regulatory agency of the United States, has published its Software Review Plan (SRP) [1] and has required safety software to be developed according to the IEEE Standard 7-4.3.2 [2]. To meet these regulatory requirements, the software used in the nuclear safety field has been ensured through the development, validation, safety analysis, and quality assurance activities throughout the entire process life cycle from the planning phase to the installation phase [3]. However, this evaluation through the development and validation process needs a lot of time and money. In addition, a variety of activities, such as the quality assurance activities are also required to improve the quality of a software. However, there are limitations to ensure that the quality is improved enough. Therefore, the effort to calculate the reliability of the software continues for a quantitative evaluation instead of a qualitative evaluation.

In this paper, we propose a reliability evaluation method for the software to be used for a specific operation of the digital controller in an NPP. After injecting random faults in the internal space of a developed controller and calculating the ability to detect the injected faults using diagnostic software, we can evaluate the software reliability of a digital controller in an NPP. In Section 2, we introduce the reliability evaluation research for a nuclear software. A specific method for software reliability evaluation of a digital controller in an NPP is explained in Section 3. In Section 4, an experiment plan is suggested. Finally, we conclude the paper in Section 5.

II. RELIABILITY EVALUATION RESEARCH FOR A NUCLEAR SOFTWARE

Active research to assess the reliability of software in the field of a nuclear power plant has only recently progressed. It has been claimed that a quantitative calculation regarding the reliability of the software is impossible owing to the assumption that the software failure rate does not increase over time unlike in electronic components. Thus, focus on the amount of testing needs to be made to ensure the reliability of the predetermined target level rather than directly calculating the software reliability. Research methods that have been tailored for this purpose thus far include the Software Reliability Growth Model (SRGM) [4] and Bayesian Belief Net [5]. However, it is premature for these research methods to be applied directly to a site because of the specificity of an NPP. The applicability of such a method may be considered only after the result is stabilized and objectively proven. Software reliability assessment methods that have been researched regarding the current status of a nuclear power plant are as follows.

A. *Software Reliability Growth Model*

The SRGM is used to establish an assumption regarding whether the software reliability will be improved when a software failure by such a defect does not occur again by removing the defects that are inherent in the event of a software failure. There are two criteria: the Root Mean Square Error (RMSE) [4] and Average Error (AE) [4].

The RMSE is a measure commonly used when dealing with the difference between one of the model predictions based on observations in the real world. It is suitable to represent the precision. Each of the difference values is also referred to as a residual, and the mean square deviation is used to synthesize the residual as a measure. These criteria may be used to measure the difference between the actual value and the predicted value. Two formulas can be expressed as follows (1)(2):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(c(k) - \hat{c}(k)\right)^2} \quad (1)$$

$$AE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{c(k)-\hat{c}(k)}{c(k)}\right| \times 100 \quad (2)$$

where n is the group number of the failure data, c(k) is the number of actual failures in each group of failure data, and ĉ(k) is the number of predicted failures. The smaller the RMSE and AE models, the more their predictive power increases.

B. Bayesian Belief Net

Bayesian Belief Net (BBN) [15] is a methodology that leads to quantitative results by calculating and applying the laws of probability including Bayes probability. It models the relevant variables in the target system through a causal relationship, expresses the dependency degree of variables as a conditional probability, and inputs several observed evidences into the BBN model generated. The BBN consists of nodes indicated as circles on the graph, arcs between nodes, and a node probability table (or conditional probability table) of each node. Nodes represent variables included in the model, the arcs indicate a causal relationship between nodes. Each node has a number of states as random variables (for example, a state of "Yes" or "No"). The sum of the probability of the state value is 1. The node probability table associated with each node determines the connection strength between nodes, and is expressed as a conditional probability for each state of the parent node.

III. Specific method for software reliability evaluation of a digital controller in an NPP

The software reliability methods we have seen thus far have been studied to apply to the software in an NPP, but there is actually no applied practice. SRGM can demonstrate that the reliability grows when failure data and the resolution case of a compete software exist. However, the adaptation data are not sufficiently secured and the BBN methodology has occupied much of the qualitative determination elements, and thus BBN methodology has a limitation in that it calculates the quantitative data. In order to overcome these disadvantages, we propose a specific method to obtain a quantitative value of the reliability of a software used in an NPP. Considerations in the proposed method are as follows.

First, the reliability evaluation formula uses the general reliability calculation method commonly used. This is the reliability calculation method for the electronic component. Applying this method to software that is not worn out may start a debate. However, we assumed that the software can also be continually exposed to potential bugs over time and that the software is also aging.

Second, random faults should be injected inside the software, and the definition for injected faults should be interpreted differently. The injected fault defined as a fault may not be recognized as a fault inside the software, and the failure weight may also be different because the injected fault has different effects on a software action.

Third, the failure rate to be used for the reliability evaluation formula should be defined. If any fault is injected in the location of the software and the fault detection coverage through the diagnostics software is calculated, the failure rate of the target software can be determined.

These issues are explained in detail as follows.

A. Reliability calculation method

A reliability is a way to express the probability that electronic components are continuously operated for a certain time. This is expressed as follows (3)(4):

$$R(t) = Pr(T \geq t) = 1 - Pr(T < t) = 1 - F(t) = 1 - \int_0^t f(t)dt \quad (3)$$

$$R(t) = 1 - F(t) = e^{-\lambda t} \quad (4)$$

F(t) is the failure cumulative distribution function and means the probability of malfunction within time t. It is expressed as follows (5)(6):

$$F(t) = Pr(T \leq t) = \int_0^t f(t)dt\,, t \geq 0 \quad (5)$$

$$F(t) = \int_0^t \lambda e^{-\lambda t}dt = 1 - e^{-\lambda t} \quad (6)$$

In addition, the *(t)* factor used in the failure cumulative distribution function of the system refers to the number of faults per unit of time. The most important factor is the failure rate *(t)* in the basic method for calculating the reliability. This is because the reliability calculation value is changed according to the number of faults in the system per unit of time. The failure rate calculation is as follows (7)(8)(9):

$$\lambda(t) = 1 - C \quad (7)$$

$$C = Pr(fault\ detected \mid fault\ existence) = {\sum FiDi}/{\sum Fi} \quad (8)$$

$$\lambda(t) = 1 - {\sum FiDi}/{\sum Fi} \quad (9)$$

There are various ways to calculate the failure rate expressed as a constant value. Among them, it is a general method that estimates the failure rate value using a probability analysis method using the test data and analysis data and calculates the reliability using the estimated values. The test data and the analysis data should be sufficient for the accuracy of the probability. However, there is a limitation in extracting the test data from the situation in which the controller is applied to the safety system and is operated. The samples also are very small, and thus it is inappropriate for use in statistics. In addition, determining the test result as a representative value of the failure rate is not rational because tests performed in the development process is not guaranteed. Random faults are injected in the software of the developed completed controller to escape the weakness, and it can then be possible to obtain the reliability of the software after calculating a failure rate using the diagnostic functions of the system.

B. Definition of SW failure in Controller

Because software within the controller in an NPP conducts the same program repeatedly, the area for the software has been limited. Thus, the definition for the fault within the controller is necessary. Because the fault occurs in the previous step of importing the system failure, even if a fault occurs, the system is not unconditionally experiencing a failure. By affecting the program or system task performing this safety function, the faults may or may not generate a system failure. For example, if even a specific area of the memory has been adhered to the value of bit 0, if the application using the memory uses the specific area as space for a constant, the integer value for the software does not change because the most upper bits remain as the value of bit 0. When the decimal value 15 is saved in the 10 bit space of integer type memory, the binary value stored in that space will be "0000001111". At this time, the upper 4 bits will always be stored as a value of zero. Although the value of the upper 4 bits fixed to a value of zero by external shock, it does not affect the safety operation of the software. These faults in the controller in an NPP should not be treated as faults. It is necessary to distinguish whether the application in the controller uses the location or not when calculating the failure of the controller in the NPP. The fault in the position where the application program is not utilized is excluded from the faults. If this fault does not affect the safety program, it is realistically difficult to detect the fault and it is also not easy to develop a diagnostic program that can detect the faults in the unused portion. In the case of implementation with a complex diagnostic algorithm, the real-time detection of the fault is not guaranteed, and the detection function may not be properly conducted because much time and a high cost are needed for its operation. The effectiveness of the fault is given depending on whether the fault can affect the operation of the software in an NPP.

C. *Fault effect factors*

There are some factors to be considered in order to determine the fault using a fault detection function. The fault coverage may be computed differently since the location, the type, and the nature of the faults are different individually. The fault factors for the software in an NPP are as follows.

Fault ∈ {type, duration, location, weight, recovery}

The type, duration, location, weight, and recovery ability are the factors for the faults. In particular, the weighting factor may have the greatest impact on the calculation of the fault detection coverage of the controller in an NPP. The recovery ability is not important in the controller in an NPP since a diversity protection system will be operated when the fault is detected. We focus on the fault detection coverage capability.

- Fault Type = stuck-0 fault，stuck-1 fault

The fault type is stuck-0 or stuck-1. A software program is operated in hardware memory and the input and output of the data are also utilized in the memory space. An action for injecting a fault occurs in the memory and the memory bit can then be stuck-0 or stuck-1. A memory fault injected in the hardware has one of the two corresponding fault types, and thus, the fault type of the target bit is determined according to a probability of 1/2.

- Fault Duration

The duration degree of a fault is one of the attributes for defining the fault. An injection fault may be lasting as a permanent fault. Another fault may be recovered to a normal state over time, although it occurs intermittently. In this study, we only consider a permanent fault and not an intermittent fault.

- Fault Location

The location of the fault is one of the attributes for quantifying it. It is important to determine whether a random fault is injected in any position. A random injection fault affects the quantification of the failure depending on whether it is located on the most significant bit or the least significant bit. The location of the fault can be defined as the weighting factor.

- Fault Weighting

Operating system software running on the safety controller in an NPP repeats the same operation, performs a calculation using the data received from the communication in repeated operations, and performs diagnostic operations. The code and data area of the accessed memory are fixed during one cycle of the application program. However, the number of accesses are different from each other. It is reasonable to assign a weight in accordance with the number of accesses because a fault in the memory space where can access frequently increases the probability, which can affect the safety operation.

IV. EXPERIMENT

An experiment for calculating the failure rate of the software in consideration of the proposed method is progressing. Until now, the memory space that a software application can access was classified according to the access count. This is shown in Figure 1.

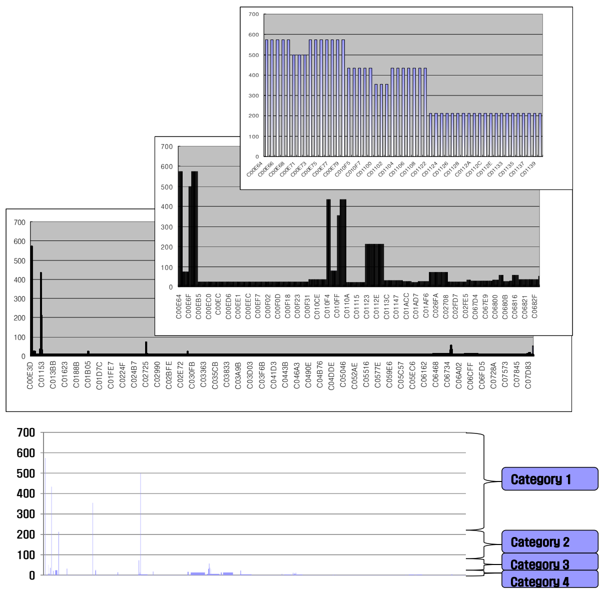
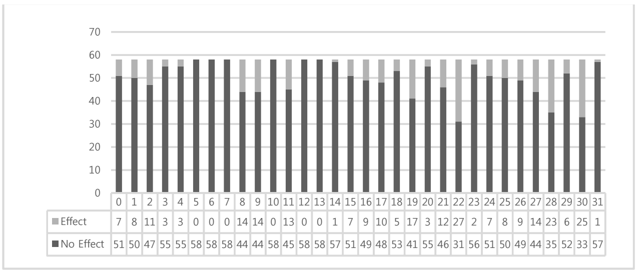


Figure 1. SW execution path and categorization.

To gain the first weighting factor related to the Fault Location, we addressed the random memory spaces that are utilized by a software program.

- Fault Injection Memory Address: 0x00C00E64 ~ 0x00C01139
- Fault Location: 0~31 bit
- Fault Type: stuck-0

Figure 2 shows the error effect statistics according to bit position in physical address.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
▨ Effect	7	8	11	3	3	0	0	0	14	14	0	13	0	0	1	7	9	10	5	17	3	12	27	2	7	8	9	14	23	6	25	1
■ No Effect	51	50	47	55	55	58	58	58	44	44	58	45	58	58	57	51	49	48	53	41	55	46	31	56	51	50	49	44	35	52	33	57

Figure 2. Error effect according to bit position of address.

Then, we can acquire the normalized weighting value using experimental sample data. Table 1 shows the normalized value depending on each bit position in address.

TABLE I. NORMALIZED VALUE OF BIT POSITION

Bit position	0	1	2	3	4	5	6	7
	8	9	10	11	12	13	14	15
	16	17	18	19	20	21	22	23
	24	25	26	27	28	29	30	31
Normalized value	-0.146	-0.013	0.389	-0.682	-0.682	-1.084	-1.084	-1.084
	0.791	0.791	-1.084	0.657	-1.084	-1.084	-0.950	-0.146
	0.121	0.255	-0.414	1.193	-0.682	0.523	2.532	-0.816
	-0.146	-0.013	0.121	0.791	1.996	-0.280	2.264	-0.950

We will estimate the software reliability using fault weighting value and the failure rate by the diagnostics software. The experimental results will be released in a future work, after the tests are completed.

V. CONCLUSION

We tried to calculate the software reliability of the controller in an NPP using a new method that differs from a traditional method. It calculates the fault detection coverage after injecting the faults into the software memory space rather than the activity through the life-cycle process. It is possible to calculate the software reliability when obtaining the failure rate and utilizing the existing calculation method. We attempt differentiation by creating a new definition of the fault, imitating the software fault using the hardware, and giving a consideration and weights for injection faults.

ACKNOWLEDGMENT

This paper was supported by the Ministry of Science, ICT (Information and Communication Technology) & Future Planning, Korea.

REFERENCES

[1] BTP-7-14, Guidance on software reviews for digital computer-based instrumentation and control system. NUREG-0800, Standard Review Plan: branch technical position 7-14, Revision 5, Nuclear Regulatory Commission.

[2] The Institute of Electrical and Electronics Engineers, Inc., “Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations,” IEEE 7-4.3.2.

[3] K. C. Kwon and M. S. Lee, “Technical Review on the Localized Digital Instrumentation and Control Systems,” Nuclear Engineering and Technology, vol. 41, no. 4, 2009, pp. 447-454.

[4] Gaurav Aggarwal and V. K Gupta, “Software Reliability Growth Model,” International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, 2014, pp. 475-479.

[5] H. S. Eom, G. Y. Park, H. G. Kang, and S. C. Jang, “Reliability assessment of a safety–critical software by using generalized Bayesian nets,” 6th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technology, Knoxville, Tennessee 2009.

[6] H. G. Kang, “An Overview of Risk quantification Issues of Digitalized Nuclear Power Plants Using Static Fault Trees,” Nuclear Engineering and Technology, vol. 41, 2009, pp. 849-858.

[7] J. Duraes and H. Madeira, “Emulation of software faults, a field data study and a practical approach,” IEEE Trans. Softw. Eng,. vol. 32, no. 11, 2006, pp. 849-867.

[8] M. C. Hsueh, T. K. Tsai, and R. KIyer, “Fault Injection Techniques and Tools,” IEEE Computer, vol. 30, no.4, April 1997, pp. 75-82.

[9] Jean arlat et al., “Fault Injection for Dependability Validation: A Methodology and Some Applications,” IEEE Trans. On Soft. Eng., vol 16, no.2, Feb 1990, pp. 166-182.

[10] G. Choi and R. Iyer, “Focus, An Experimental Environment for Fault Sensitivity Analysis,” IEEE Trans. On Computers, vol.41, no.12, December 1992, pp. 1515-1526.

[11] Y. Yu, “A perspective on the state of Research on Fault injection techniques,” Research Report, University of Virginia, May 2001.

[12] PATENT, “Fault mode apparatus and method using software,” 10-1222349, The Korean Intellectual Property Office, 2013.

[13] H. Madeira, D. Costa, and M. Vieira, “On the emulation of software faults by software faults by software fault injection,” Proceedings of International Conference on Dependable Systems and Networks, 2000, pp. 417-426.

[14] S. Richter and J. Wittig, “Verification and Validation Process for Safety I&C Systems,” Nuclear Plant Journal, May-June, 2003, pp. 36-40.

[15] B.A. Gran and A. Helminen, “The BBN methodology: progress report and future work. OECD Halden Reactor Project,” HWR-693, 2002.