Transactions of the Korean Nuclear Society Autumn Meeting
Gyeongju, Korea, October 27-28, 2016

A quantitative calculation for software reliability evaluation[i]

Young-Jun Lee, Jang-Soo Lee
Korea Atomic Energy Research Institute
yjlee426@kaeri.re.kr, jslee@kaeri.re.kr

1. Introduction

Nuclear Regulatory Commission(NRC) 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 ensure the safety of software used in a Nuclear Power Plant (NPP). 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 quantitative calculation 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

2. Quantitative Calculation Method

We propose a 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

2.1. 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 (1)(2):

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

$$R(t) = 1 - F(t) = e^{-\lambda t} \quad (2)$$

F(t) is the failure cumulative distribution function and means the probability of malfunction within time t. It is expressed as follows (3)(4):

$$F(t) = Pr(T \leq t) = \int_0^t f(t)dt\,, t \geq 0 \quad (3)$$

$$F(t) = \int_0^t \lambda e^{-\lambda t}dt = 1 - e^{-\lambda t} \quad (4)$$

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 (5)(6)(7):

$$\lambda(t) = 1 - C \quad (5)$$

$$C = Pr(fault\ detected \mid fault\ existence) = {\sum FiDi}/{\sum Fi} \quad (6)$$

$$\lambda(t) = 1 - {\sum FiDi}/{\sum Fi} \quad (7)$$

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. 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.

2.2. 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.

2.3. 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.

- 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
  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.

3. Experiment

An experiment for calculating the failure rate of the software in consideration of the proposed method is as follows.

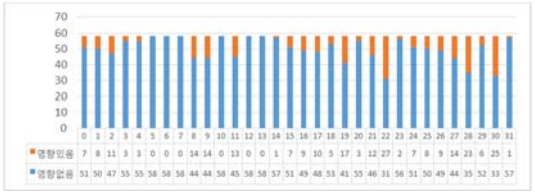

Figure1. Error effect statistics through 0~31 bit

Figure2. Weight ratio according to bit position

Figure3. Access count of each address

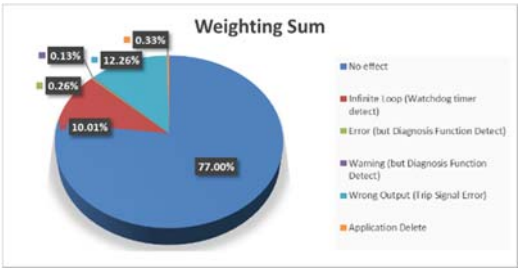

Figure4. Fault detection coverage using weighting values

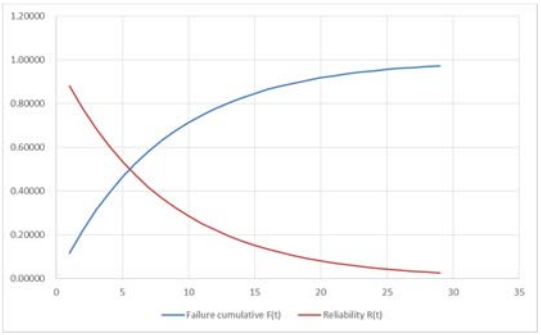

Figure5. Relation between target software and failure cumulative distribution

4. 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

- This paper was supported by the Ministry of Science, ICT (Information and Communication Technology) & Future Planning
- This paper is an extended version of "A specific method for software reliability of digital controller in nuclear power plant" which was presented in FASSI2016

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