Safety case of digital protection systems focusing on safety-critical software failure in NPPs

*Hee Eun Kim [a], Han Sung Son [b], Hyun Gook Kang [c,]**
[a] Department of Nuclear and Quantum Engineering, Korea Advanced Institute of Science and Technology, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon 34141
[b] Department of Gaming, Joongbu University, 201 Daehak-ro, Chubu-myeon, Geumsan-gun, Chungnam, 312-702,
[c] Department of Aerospace and Nuclear Engineering Rensselaer Polytechnic Institute, Troy, NY 12180, USA

**Corresponding author: kangh6@rpi.edu*

1. Introduction

The safety systems of nuclear power plants are manipulated by the digital instrumentation and control systems, which provide the control and monitoring functions of the various and diverse components and equipment that are essential to maintaining safe operation.

Software failure can be defined as not successfully performing a specified/intended function or performing unintended actions [1]. It is known that software failure is one of the most important features of digital systems. A study [2] also identified that the estimation of software failure probability is an important factor. The sensitivity study in the study on the digital reactor protection system (RPS) showed the relationship of system unavailability and software failure probability. RPS unavailability causes the failure of safety signal generation in the event of an accident and finally would harm accident mitigation process. In this paper, brief description of software safety case development will be discussed.

2. Causes of software failure

In this section, causes of software failure are categorized into three categories, based on the software failure model suggested by Chu, et al. [3].

2.1 Internal cause

Software design fault can be introduced during its software life cycle. Software failures can occur when certain input values or input sequences (trajectory) are given. The inputs can interact with the error crystal or an internal state of the digital system to trigger a fault that was introduced into the software during the software lifecycle [1].

2.2 Intentional external causes

Malicious attack from a hacker can be one of the causes of software failures. Since a malicious cyber-attack is intentional, the frequency or probability of cyber-attack cannot be estimated quantitatively. The scale and scope of cyber-attack may vary on the attacker's capability and knowledge on NPP. It can be assumed that cyber-attack on any component or system is possible. Analysis needs to be focused on the failure of safety function or spurious trip, rather than information theft.

2.3 Unintentional external causes

Other causes can be categorized into this type, for example, failure of the supporting system and overall context. The supporting system includes other software, computer hardware, electric power, and possibly a system, such as heating.

3. Safety cases of the RPS software failure

This section presents a simple structure of the RPS software safety case and possible way of obtaining corresponding evidence.

3.1 Safety case

The safety case is a structured argument, supported by a body of evidence that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given operating environment [4]. Safety cases for RPS software can be developed, and corresponding evidence can be provided by test result to guarantee digital system safety. General structure of safety case contains a claim, sub-claims, and arguments for the claims with supporting evidence.

3.2. Development of Safety cases for the RPS software

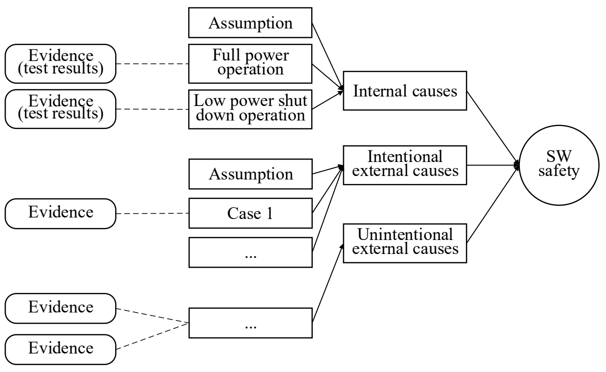

Fig. 1. Safety case of RPS software

In case of the digital protection system, the overall claim is that the software of the system is safe. Since the mechanisms of software failure are different according to the types of the causes, the claim can be divided into three sub-claims (Fig. 1).

Software failure could be treated based on a probabilistic method because of the randomness of the input sequences, if a specific application is concerned [5]. Therefore, the result of software test can be the evidence for the internal cause of the software failure. The method to determine exhaustive test set has been described in the other article of the authors [6].

In case of intentional external causes, a malicious cyber-attack is deliberate, so the frequency or probability, and the scale and scope of cyber-attack cannot be estimated quantitatively. To develop safety case, critical digital assets need to be determined according to the core damage scenario. The core damage scenarios and corresponding penetration test results on the critical digital assets can be the evidence for the intentional external cause of the software failure. Scenarios derived from the safety assessment model can be utilized as evidence [7].

For the unintentional external causes, evidence needs to incorporate the effect of the operating system and the hardware, etc. Many unknown factors for the failure such as software-hardware interactions and influence of supporting software failures need to be identified first, and the safety cases can be defined for them. Risks need to be identified to be managed and the failure modes could be obtained from software tests including hardware platform, or operational history.

3.3. Test to obtain evidence of unintentional external cause

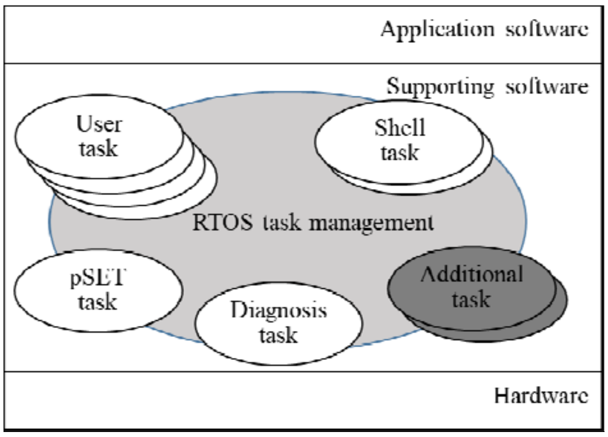

Fig. 2. Accelerated testing for task management

Accelerated testing can be performed on real-time operating system (RTOS) software because it is simple and highly reliable. Acceleration factor can be determined according to the purpose of the test, for example, use rate and level of stress. Since the RTOS is supporting software and the functions related to the reliable execution of application software needs to be tested, so the acceleration factor can be selected according to the functions of the RTOS. In case of task management function, increase of use rate can be the accomplished by adding tasks (Fig. 2).

Testing on a real hardware platform might be appropriate to reflect realistic environmental impacts. Possible failure modes might be obtained from simulation so that that testing effort can be reduced. If sufficient amount of data is collected, data estimation by extrapolation might be possible. However, it is an empirical model, and detailed failure mechanisms are not clearly identified, so it is an alternative way of data estimation.

4. Conclusions

RPS software safety case can be defined based on each type of software failure causes in consideration of failure characteristics. If suggested evidence of each claim is provided, it can be said that the risk from RPS software is sufficiently low. Safety case considering software environment needs to be developed further.

REFERENCES

[1] T. L. Chu, et al., Review of quantitative software reliability methods. No. BNL--94047-2010. Brookhaven national laboratory, 2010.

[2] H. Kang, and T. Sung, An analysis of safety-critical digital systems for risk-informed design, Reliability Engineering & System Safety Vol. 78, No. 3, pp. 307-314, 2002

[3] T. L. Chu, et al. A review of software-induced failure experience. No. BNL-NUREG-77124-2006-CP. Brookhaven National Laboratory, 2006.

[4] UK Def., Safety management requirements for defence systems. Part 1, 1996.

[5] H. Kang, et al., Input-profile-based software failure probability quantification for safety signal generation systems, Reliability Engineering & System Safety, Vol. 94 No.10, pp. 1542-1546, 2009.

[6] H. E. Kim, et al., Input-domain Software Testing for Failure Probability Estimation of Safety-Critical Applications in Consideration of Past Input Sequence, IEEE Access, Vol. 6, pp. 8440-8451, 2018.

[7] H. E. Kim, H. S. Son, J. Kim, & H. G Kang, Systematic development of scenarios caused by cyber-attack-induced human errors in nuclear power plants, Reliability Engineering & System Safety, Vol 167, pp. 290-301, 2017.