Quantitative Assessment Methodology of Safety-Critical Software Reliability in Digital I&C Systems

Seung-Cheol Jang*, Sung-Min Shin, Jin-Kyun Park and Jong-Gyun Choi
Korea Atomic Energy Research Institute, 111, Daedeok-daero 989beon-gil, Yuseong-gu, Daejeon, 34057
**Corresponding author: scjang@kaeri.re.kr*

Keywords : Digital I&C System, Safety-Critical Software, Software Reliability, Quantitative Assessment, Probabilistic Safety Assessment (PSA)

1. Introduction

Digital systems offer several advantages over analog systems, such as zero drift, high data processing capability, and design flexibility, leading to their expanded introduction into domestic nuclear power plant safety systems since the late 1990s. However, various safety issues arise from the software (SW) control characteristics in digital nuclear power plant systems, and SW failure is particularly important as one of the major common cause failure (CCF) elements of the system. Although the U.S. National Research Council (NRC) recommended in 1997 that SW failure impact assessment models be included in digital I&C PSA[1], quantitative reliability assessment of nuclear safety-critical SW remains an ongoing technological issue without an approved methodology in the nuclear industry. The absence of such a methodology acts as a bottleneck for the Risk-Informed Decision Making (RIDM) regulatory framework.

The objective of this paper is to propose an applicable quantitative SW reliability assessment framework for safety-critical SW in nuclear digital safety systems [2].

2. Characteristics of Safety-Critical Software in Nuclear Power Plants

2.1 Software Failure Mechanisms and Characteristics

SW failures can be understood through the relationship between SW error, fault (or defect), and failure as shown in Fig.1[3].

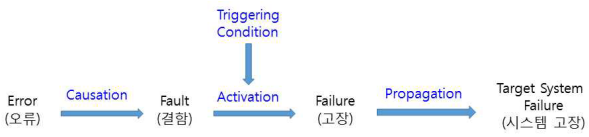

Fig. 1. Relationship among SW error, Fault and Failure

An error is the cause of a fault, while a fault is a defective state in a system or SW caused by a SW error. A failure is the result of a fault activated by a trigger, which is defined as a specific triggering event or condition that causes a system failure due to a latent SW fault. SW failures are events that occur when undetected latent faults are activated by randomly occurring triggers during the SW's lifetime. This randomness of triggers provides the basis for probabilistically addressing SW reliability.

Unlike hardware failures, which result from manufacturing defects, aging, wear, or environmental impacts, SW is not subject to aging or wear and is associated with inherent errors from the development phase. SW failures are systematic due to the repetition of the same triggering conditions, but the triggering conditions themselves occur randomly. SW failure triggering conditions can arise from incorrect analysis during the requirements definition phase, changes in the operating environment, or undetected latent design errors due to inadequate design.

2.2 Existing Software Reliability Assessment Methods and Limitations

While SW reliability modeling in PSA remains an unresolved issue, there is general consensus that SW can fail, its failures can be treated probabilistically, and it is meaningful to use SW failure rates and probabilities in digital system reliability models[4]. Various quantitative SW reliability assessment methods have been proposed, including reliability growth models, Bayesian Belief Networks (BBN), SW matrix-based methods, and test-based methods. However, these methods have not been widely adopted in nuclear PSA for several reasons. The most fundamental reason is that many of these models (e.g., reliability growth models, metrics-based models) are designed to estimate the number of remaining faults in the software. This premise conflicts with the reality of nuclear safety systems: software is not permitted to be deployed if it is known to contain faults. Consequently, PSA practitioners tend to rely on engineering judgment, such as screening out SW failures due to their perceived minor contribution, using screening values based on Safety Integrity Level (SIL), or expert judgment.

3. Quantitative Software Reliability Assessment Method for Nuclear Power Plants

The proposed Quantitative Software Reliability Model (QSRM) provides a structured, top-down approach to quantify software reliability by decomposing the complex problem into manageable components. The framework is based on the principle

that the total probability of software failure is the sum of failure probabilities across a set of mutually exclusive and collectively exhaustive (MECE) operating environments. Its Bayesian foundation allows for the systematic integration of diverse evidence types.

3.1 Framework Structure and Theoretical Basis

The model defines the total software operating environment(Ω) as a collection of distinct environments (Ei), where each environment represents a specific operational context (e.g., normal operation, testing, abnormal conditions). The relationship is visualized in Fig. 2.

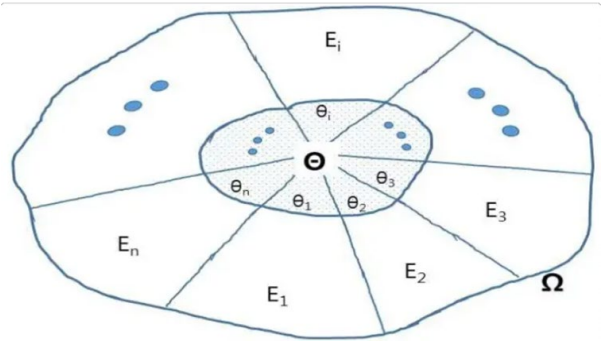

Fig. 2. Relationship between software operating environment (Ei) and failure events (θi).

The overall software failure probability, P(Θ), is calculated by summing the conditional failure probabilities across all MECE operating environments. The fundamental model is expressed as:

$$P(\theta) = \sum_{i=1}^{n} P(E_i) \cdot P(\theta | E_i) \quad \text{(Eq.1)}$$

Furthermore, safety-critical systems incorporate fail-safe design features (e.g., watchdog timers, self-diagnostics). A software failure only leads to a Target System Failure (TSF) if these protective features also fail. This is incorporated into the model via a fail-safe failure probability, k(Ei). The final probability of a TSF is given by:

$$P(TSF) = \sum_{i=1}^{n} k(E_i) \cdot P(E_i) \cdot P(\Theta | E_i) \quad \text{(Eq.2)}$$

3.2 Quantification of Model Components

The QSRM framework requires the quantification of three key parameters:

- ✓ Probability of Operating Environment, $P(E_i)$: This term represents the likelihood of the system being in a specific operating environment *i*. It can be estimated from the time fraction the system operates in that environment, based on operational data or plant-specific analysis.
- ✓ Conditional Software Failure Probability, $P(\Theta|E_i)$: This is the probability of a software failure given the system is in environment E_i. Due to the scarcity of direct failure data, a Bayesian updating approach is proposed. The posterior probability is proportional to the product of the prior probability and the likelihood of observing evidence:

$$P\{\theta | d(E_i)\} \propto P(\theta) \cdot P\{d(E_i) | \theta\} \quad \text{(Eq.3)}$$

  - Prior Distribution, $P(\theta)$: This is an initial belief of SW reliability and can be derived from established guidelines, such as the failure probability ranges associated with Safety Integrity Levels (SIL) in IEC 61508 [5].
  - Likelihood, $P\{d(Ei)|\theta\}$: This is derived from evidence d(Ei), which can include operational data (e.g., number of failures per demand/time) or results from systematic testing.
- ✓ Fail-Safe Failure Probability, $k(E_i)$: This is the probability that the system's built-in safety and diagnostic features fail to detect or mitigate a software failure. This value can be quantitatively estimated through methods like fault injection experiments, which measure the coverage of self-diagnostic functions.

4. Case Study: Application to a Digital Plant Protection System (DPPS)

To illustrate the QSRM framework, a case study is performed on a hypothetical DPPS. The objective is to calculate the demand-based failure probability of the system's safety-critical software.

Step 1: Define Operating Environments (E_i)

The software's operating environment is partitioned into three MECE states:

- ✓ E_1 (Normal Operation): Continuous execution path for processing normal plant parameters. It is assumed to be thoroughly verified and validated (V&V) during development.
- ✓ E_2 (Anticipated Abnormal Conditions): Infrequent but expected events like plant startup, single-channel testing, or system initialization. V&V is assumed to have been performed for these scenarios.
- ✓ E3 (Unverified Abnormal Conditions): Unforeseen plant states or configurations not covered during V&V. These paths are most likely to trigger latent faults.

Step 2: Quantify P(Ei) and k(Ei)

Based on assumed operational profiles, the time fractions and fail-safe failure probability are assigned (see Table 1):

Table 1. P(Ei) and k(Ei) Values

Parameter	Value	Basis
$P(E_1)$	0.85	Assumed time fraction for normal operation.
$P(E_2)$	0.10	Assumed time fraction for anticipated abnormal states.
$P(E_3)$	0.05	Assumed time fraction for unverified abnormal states.
$k(E_i)$	0.1	A conservative value assumed for all environments, based on typical results from fault injection experiments showing the probability of failure detection failure.

Step 3: Bayesian Update for $P(\Theta \mid Ei)$

The conditional failure probabilities are estimated using Bayesian updating (see Table 2 and Fig.3):

Table 2. The Results of Bayesian Update

Environment	Prior Distribution (Mean Failure Prob.)	Likelihood (Evidence)	Posterior Distribution (Mean Failure Prob.)
E_1	1.0E-4 (from SIL 4)	Extensive testing shows near-perfect integrity (e.g., 0 failures in 1.35E+11 tests[6]).	1.0E-6 /demand (Assumed minimum value)
E_2	1.0E-3 (from SIL 3)	Assumed 0 failures in 1,000 demands.	6.94E-4 /demand (Calculated via Bayesian update[7])
E_3	1.0E-2 (from SIL 2)	Assumed 0 failures in 1,000 demands.	2.68E-3 /demand (Calculated via Bayesian update[7])

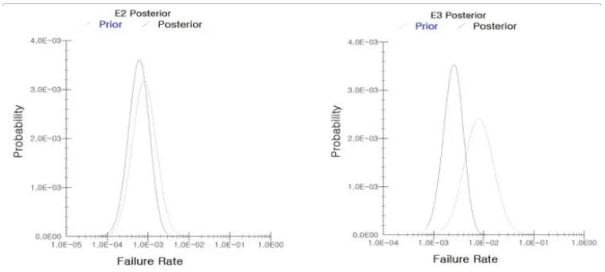

Fig. 3. Posterior Distributions for E2 & E3 using BURD code[7]

Step 4: Calculate Final Target System Failure (TSF) Probability

Using the comprehensive formula, the final P(TSF) is calculated:

$$
\begin{aligned}
P(TSF) &= [k(E_1) \cdot P(E_1) \cdot P(\Theta|E_1)] \\
&\quad + [k(E_2) \cdot P(E_2) \cdot P(\Theta|E_2)] \\
&\quad + [k(E_3) \cdot P(E_3) \cdot P(\Theta|E_3)] \\
&= [0.1 \cdot 0.85 \cdot 1.0E\text{-}6] \\
&\quad + [0.1 \cdot 0.10 \cdot 6.94E\text{-}4] \\
&\quad + [0.1 \cdot 0.05 \cdot 2.68E\text{-}3] \\
&= 8.5E\text{-}8 + 6.94E\text{-}6 + 1.34E\text{-}5 \\
&= 2.04E\text{-}5/\text{demand}
\end{aligned}
$$

The final result of 2.04E-5/demand is highly dependent on the input parameters. A case study underscores the importance of robust V&V processes aimed at reducing the size of the "unverified" state space and developing comprehensive self-diagnostics (reducing $k(E_i)$). The QSRM framework makes these dependencies explicit and quantifiable.

5. Conclusions

The Quantitative Software Reliability Model (QSRM) presented in this paper offers a systematic, flexible, and transparent framework for assessing the reliability of safety-critical software in nuclear digital I&C systems. By leveraging a Bayesian approach, it provides a robust mechanism to integrate various forms of evidence—from design-level information like SIL ratings to empirical data from operations and testing. The decomposition of the problem by operating environments allows for a more nuanced and realistic analysis.

The case study demonstrates that the framework is practical and yields transparent, reproducible results. The significance of the QSRM lies in its ability to provide an analytical structure for combining existing reliability assessment methods, thereby producing a credible failure probability estimate for PSA models. While the model's accuracy is contingent on the quality of the input data and the validity of the underlying assumptions, it represents a significant step forward from purely judgmental approaches. It provides a defensible basis for regulatory review and enhances the rigor of safety cases for digital systems. Future work should focus on refining the methods for data collection, particularly for estimating $P(E_i)$ from operational data and $k(E_i)$ through advanced testing techniques, and on applying the framework to a wider range of real-world digital safety systems in the nuclear industry.

REFERENCES

[1] National Research Council, Digital Instrumentation and Control Systems in Nuclear Power Plants: Safety and Reliability Issues. Final Report, Washington, D.C.: National Academy Press, 1997.
[2] Seung-Cheol Jang. *et. al*., A Quantitative Reliability Assessment Method for Safety-Critical Software in Digital I&C Systems, Technical Report, KAERI/TR-9857/2023 (2023-NG-0002-0054), KAERI, 2024.
[3] IAEA, Dependability Assessment of Software for Safety Instrumentation and Control Systems at Nuclear Power Plants, IAEA Series No. NP-T-3.27, IAEA, 2018.
[4] T.-L. Chu, *et. al*., Workshop on Philosophical Basis for Incorporating Software Failures in a Probabilistic Risk Assessment. BNL-90571-2009-IR, New York: Brookhaven National Laboratory, 2009.
[5] International Electrotechnical Commission, Functional Safety of Electrical/Electronic/Programmable Safety-Related Systems, Part 1: General requirements. IEC 61508-1, 2010.
[6] KHNP, Evaluation of Human Error Probability and Safety Software Reliability in a Digital Environment (Final Report). TR A11NJ10, 2019 (In Korean).
[7] Seung-Cheol Jang, *et. al*., BURD (Bayesian Update for Reliability Data) Program (Ver. 4.01). Registration No. 2001-01-12-5517, KAERI, 2001.