Practicality for Software Hazard Analysis for Nuclear Safety I&C System

Yong-Ho Kim *, Kwon-Ki Moon, Young-Woo Chang, Soo-hyun Jeong
KEPCO E&C, Inc., 989-111, Daedeok-dae-ro, Yuseong-gu, Daejeon, 34057, Korea
**Corresponding author: yongho@kepco-enc.com*

1. Introduction

System is a framework of our world. The word "system" used in engineering is also used in various fields as a generalized term (e.g., social system, legal system, and transport system, etc.).

We are using the concept of system safety in engineering. It is difficult to make any system perfectly safe and probably a complete system may not easily be achieved.

The standard definition of a system from MIL-STD-882E is: "The organization of hardware, software, material, facilities, personnel, data, and services needed to perform a designated function within a stated environment with specified results." From the perspective of the system safety engineer and the hazard analysis process, software is considered as a subsystem [1].

Regarding hazard analysis, to date, methods for identifying software failures and determining their effects is still a research problem, especially since there is no clear industry and regulatory consensus on the meaning of "software failure." [2]

Since the success of software development is based on rigorous test of hardware and software, it is necessary to check the balance between software test and hardware test, and in terms of efficiency.

2. Hazard Theory

In order to perform hazard analysis, a basic understanding of hazards and mishaps is required.

A hazard is comprised of the following three basic components [3]:

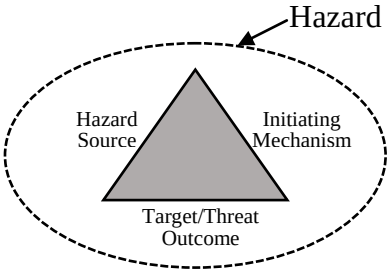

Fig. 1. Hazard triangle.

1. Hazard Source (HS)
   This is the basic hazardous resource creating the impetus for the hazard, typically a hazardous energy source such as explosives being used in the system [3].
2. Initiating Mechanism (IM)
   This is the trigger or initiator event(s) causing the hazard to occur. The IM causes actualization or transformation of the hazard from a dormant state to an active mishap state [3].
3. Target and Threat Outcome (TTO)
   This is the person or thing that is vulnerable (target), the threat to that target (threat), and the resulting severity outcome (outcome) when the mishap event occurs. The outcome is the expected consequential damage and loss [3].

Removal of any one of the triangle sides and the hazard is eliminated because it is no longer able to produce a mishap (i.e., the triangle is incomplete) [3].

In particular, removing IM is one of the easier ways and the most effective means for detecting IMs is the software test.

3. Software Hazard Analysis and Results

According to (system) hazard theory above, we categorized software hazard components using software hazard. Fig. 2 shows how this software hazard is divided into the three basic software hazard components.

Software hazard	**Software hazard components**		
In the system with potential error, operator could input a numerical value below acceptable range into the system by mistake. As a result unexpected output signal has occurred.	HS	In the system with potential error	Causal Factors
	IM	operator could input a numerical value below acceptable range into the system	
	IM	by mistake	
	TTO	As a result unexpected output signal has occurred.	Out-come

Fig. 2. Example of software hazard components.

Per hazard theory above and the guidance of NUREG/CR-6430 [4] software hazard analysis for nuclear safety I&C system was performed to evaluate the potential impact of plausible software failures on identified hazards. Only the software portion of the

system is considered. In particular, it is assumed that the computer hardware operates without failure. As a consequence of the above assumptions, the scope of this analysis is concentrated on two key issues:

- If the software operates correctly, what is the potential effect on system hazards?
- If the software operates incorrectly, what is the potential effect on system hazards?

Examples of the software hazard analysis process and input/output are as follows:

Table. 1. Software hazard analysis – input/output

Phase	Input	Procedure	Output
Requirements Phase **(Analytic target: modules)**	PHL* PHA* SysRS* SRS* Requirements Phase RTM*	Deviations are defined as hazardous states that are listed in the PHL. The causes and consequences for each deviation are analyzed and then the results and/or recommendations are recorded.	A list of software hazard. An analysis of the impact on hazards of the software when it operates correctly or incorrectly with respect to meeting each requirement.
Design phase **(Analytic target: subroutines)**	PHL PHA SRS SDD Design Phase RTM	Deviations are defined as hazardous states that are listed in the PHL for nuclear safety I&C system.	An analysis of the impact on hazards of the software when the specified software design is used.
Implementation and Test phase **(Analytic target: untested items)**	PHL PHA SRS SDD Code TC and TP Implementation and Test Phase RTM	Deviations are defined as hazardous states that are listed in the PHL for nuclear safety I&C system.	The product of the implementation and test phase HA contains the information for further verification of the code and software testing (module, unit, etc.) if there are any.

*HA: Hazard Analysis, PHA: Preliminary Hazard Analysis
PHL: Preliminary Hazard List
SRS: Software Requirements Specification
SysRS: System Requirements Specification
TC: Test Cases, TP: Test Procedure
RTM: Requirements Traceability Matrix

The software hazard analysis refers to the next Table.

Table. 2. Software hazard analysis table sample

Phase Guide Phrases [IM]	Function Level Deviation [HS]	Causes [IM]	Consequ-ences [TTO]	Safe-guards	Hazard Control Verification Method
Require-ments Phase Numerical value within range, but wrong	Data used for the following functions are within range, but wrong: [Modules] MOVAVG, W2IL10, IL2W10, R2W10, FLOWMOD1, TRIPBUF1, UPDTMD1A, POWRMOD1, STATMOD1, PFMOD1, PFSNAP1, ... (The rest is omitted)	Entry error; Programming error; Conversion error.	Undetected problem in single channel could leave failed channel in service. Consistent problem in all channels could give misleading indication, delay/prevent system trip.	4 channel redundancy; channel alarm on failures in the following: addressable constants updated (automatically) periodically from OM and MTP (with CRC); automatic periodic CRC calculation of code and fixed data; ..... (The rest is omitted)	Software testing; document and code reviews; administrative control of changes; error reporting and tracking; periodic tests; Cross channel comparison.
Design phase Numerical value within range, but wrong	Data used for the following subroutines are within range, but wrong: [subroutines] SET_IO_CONFIGURATION READ_GLOBAL_MEMORY FLOW, UPDATE, TRIPSEQ, TRIP BUFFFERS, WRITE_GLOBAL_MEMORY, MOVE_NETWORK_OUTPUTS,	Same as above	Same as above	Same as above	Same as above
	..... (The rest is omitted)				
Implemen-tation and Test phase Numerical value within range, but wrong	'Test 2 – Inputs and Outputs' does not test this deviation 'Test 4 – CEAP/CCP Inputs' does not test this deviation 'Test 5 – COPP Static Input' does not test this deviation 'Test 6 – COPP Dynamic Input' does not test this deviation 'Test 10 – CEAP Snapshot' does not test this deviation 'Test 11 – COPP Trip Buffer' does not test this deviation 'Test 17 – RPC Test' does not test this deviation 'Test 19 – CWP Test' does not test this deviation ..... (The rest is omitted)	Same as above	Same as above	Same as above	Same as above

The software hazard analysis confirmed that with adequate document review, code inspection and software testing, the nuclear safety I&C system software can perform its protection functions as required.

The analysis on lessons learned demonstrates that the nuclear safety I&C system software provides low probability of creating hazards even when it fails. Therefore the nuclear safety I&C system design is capable of performing its protective functions with high reliability.

It should be noted in Table 2 (e.g., Safeguards, Hazard Control Verification Method), although hazard analysis is performed, what reduce or mitigate hazards are the task of implementation and test phase works such as redundancy, code review, software test and integrated test. That is, the major role for measures is the realistic system configuration, extensive test action and diagnostics, etc.

Also, in the latest report, it is described that the extensive testing of the integrated hardware/software system in its native environment still remains the most useful approach to reducing software hazards due to system changes. Because the current ability remains weak in using inspections to detect errors introduced in requirements, design and implementation, software development success can depend strongly on extensive testing, diagnostics and repair of the source code [2].

Faults that have not been introduced during the development process of the complex logic or that have been removed during verification and validation will never appear in use [5].

In other words, the direction of the services centralized could be considered through the experience and portion of the mutual role of hazard analysis and test.

4. Conclusions

Lessons learned and experience from similar systems are important for the work of hazard analysis. No major hazard has been issued for the software developed and verified in Korean NPPs. In addition to hazard analysis, software development, and verification and validation were thoroughly performed.

It is reasonable that the test implementation including the development of the test case, stress and abnormal conditions, error recovery situations, and high risk hazardous situations play a key role in detecting and preventing software faults.

As a conclusion, while maintaining the current hazard analysis level, rigorous test is a more recommended approach.

REFERENCES

[1] MIL-STD-882E, "DOD Standard Practice - System Safety", 2012.
[2] EPRI, "Hazard Analysis Methods for Digital Instrumentation and Control Systems", Technical Report, 2013.
[3] Clifton A. Ericson, "Hazard Analysis Techniques for system safety", 2015.
[4] NUREG/CR-6430, "Software Safety Hazard Analysis", 1996.
[5] NUREG/IA-0254, "Suitability of Fault Modes and Effects Analysis for Regulatory Assurance of Complex Logic in Digital Instrumentation and Control System", 2011.