Transactions of the Korean Nuclear Society Spring Meeting
Jeju, Korea, May 22, 2009

Application of Software Safety Analysis Methods

Gee-Yong Park [a*], Sup Hur [a], Se-Woo. Cheon [a], Dong H. Kim [a], Dong Y. Lee [a], Kee C. Kwon [a], Sungjin Lee [b], Youngho Koo [b]
[a] *I&C and Human Factors Division, KAERI, 1045 Daedeok-daero, Yuseong, Daejeon, 305-353, Korea*
[b] *Nuclear Safety System Team, Doosan Heavy Industries & Construction Co., Ltd., Korea*
[*]*Corresponding author: gypark@kaeri.re.kr*

1. Introduction

A fully digitalized reactor protection system, which is called the IDiPS-RPS, was developed through the KNICS project. The IDiPS-RPS has four redundant and separated channels. Each channel is mainly composed of a group of bistable processors which redundantly compare process variables with their corresponding setpoints and a group of coincidence processors that generate a final trip signal when a trip condition is satisfied. Each channel also contains a test processor called the ATIP and a display and command processor called the COM. All the functions were implemented in software. During the development of the safety software, various software safety analysis methods were applied, in parallel to the verification & validation (V&V) activities, along the software development lifecycle. The software safety analysis methods employed were the software hazard and operability (Software HAZOP) study, the software fault tree analysis (Software FTA), and the software failure modes and effects analysis (Software FMEA).

2. Software Safety Analysis Methods

2.1 Software HAZOP

The software HAZOP is a main technique used in the activities of software safety analysis for the IDiPS-RPS software. It has been applied from the software requirement specifications to the implemented codes among the software development lifecycle.

The software HAZOP has some distinguishing features in that the quantity to be deviated is not quantitative but qualitative, i.e., the deviation quantities are the software functional characteristics such as accuracy, capacity, functionality, reliability, robustness, security, and safety [1]. Moreover, with the software functional characteristics as a deviation quantity, guide phrases rather than guidewords are devised for a systematic deviation and a checklist is made up based on these guide phrases. The use of the guide phrases was originally proposed by the LLNL in the form of the NUREG report [2]. Based on this NUREG report, the guide phrases applicable to the requirements, design, and implementation phases for the IDiPS-RPS software were devised carefully.

The reason for selecting this type of the software HAZOP is due to the KNICS project's policy that the proven technology (or a technology that has drawn a consensus in the nuclear fields) has the highest priority for the application.

The software HAZOP is performed to identify some software defects that can induce one of the software-contributable system hazards when a certain deviation for each functional characteristic is applied to the software system. The software-contributable system hazards are presented in Table 1 for the IDiPS-RPS.

Table I: Software-Contributable Hazards for IDiPS RPS

Item No.	Hazards	Criticality Level
1	IDiPS cannot generate a trip signal when a trip condition for a process variable is satisfied.	4
2	IDiPS generates a trip signal when it should not generate a trip signal.	3
3	IDiPS cannot send qualified information of its operating status to the main control room.	2

Table 2 presents one example of guide phrases for corresponding functional characteristics for application to a software requirements specification.

Table II: Some of Guide Phrases for Application at Requirements Phase

Attribute	Guide Phrases
Accuracy	Wrong variable type
Accuracy	Wrong variable name
Capacity	Message volume is erratic
Capacity	Untimely operator action
Functionality	Function is not carried out as specified (for each mode of operation)
Functionality	Function is executed in incorrect operational mode
Reliability	Software fault tolerance requirements (if any) are not met
Robustness	Software fails in the presence of unexpected input data
Safety	S/W causes system to move to a hazardous state
Security	Unauthorized person has access to S/W system

The software HAZOP analysis investigates all the safety-critical, trip-functioning software requirements, design specifications, and code to determine whether there is a defect connected to the hazards in Table 1, by applying iteratively all the items in the deviation checklist to each software sub-system or module.

The software HAZOP method has an advantage of the applicability to any form of documents or programs. It can be applied to a document written by natural language and also applicable to a formal description or a software code. But, this requires a considerably large amount of time and efforts of the software HAZOP members. The software HAZOP was proven to be useful in identifying a software hazard affecting the system safety and availability, especially at the requirements and design phases [3].

2.2 Software FTA

The software FTA was applied to the detailed design descriptions at the design phase and the source code at the implementation phase. The trip-functioning software modules of the IDiPS-RPS were implemented by a function block diagram (FBD) that is provided by the POSAFE-Q PLC. The software FTA was applied to only the software design/implemented modules that were identified to be defective from the software HAZOP analysis. It searches a defect's cause and location that can induce the most significant system hazard as in the first item (Item No. 1) in Table 1.

The purpose of the application of the software FTA at the design/implementation phases is to compensate for the software HAZOP. Though an FTA was proposed as a software safety analysis method at the appendix of NUREG/CR-6101 [4], the application of the software FTA to the safety software is rare in the nuclear fields but this method with a systematic analysis procedure has been successfully applied to non-nuclear software systems [5].

In order to systematically perform the software FTA analysis, the two basic frameworks were performed a priori. One is the identification of the interface points between a software module and the IDiPS-RPS system. And the other is the establishment of the fault tree template. The fault tree template was constructed for each function block in the FBD and is drawn from the intrinsic failure events of a function block. Considering the facts described above, the software FTA is more suited for applications at the design or implementation phase rather than at the requirements phase because the software structure is defined in more detail in the software design architecture and the implemented code can provide more concrete software unit elements.

In contrast to the software HAZOP analysis where a forward broad-thinking analysis is provided, the software FTA method is a backward step-by-step local analysis method. It begins from the top node which represents an unsafe state and searches for the causes of the top event through logical paths of an FBD module, up to its inputs. Moreover, the SFTA is usually performed by an individual expert rather than through a meeting of analysis team members. From the application results of the software FTA [6], the software FTA based on the fault tree template could delicately search for a local defect or some logical error. But it is difficult for the software FTA to apply to all the software modules because of its complex structure and time-consuming work.

2.3 Software FMEA

This method was applied to a part of the source code in the ATIP that performs subsidiary functions relating to various tests and interface functions.

A failure mode template was devised for a systematic analysis for an FBD-implemented code and, in this case, one single failure mode template (the fault tree template of the software FTA consists of many sub-templates) was applied to all the FBD modules. Table 3 presents a simple failure mode template for the application to an FBD code.

Table III: Failure-Mode Template for Software FMEA

ITEM	FAILURE MODES
Function	Omission
	Incorrect Realization
	Unintended Addition
	Function Interaction
Input	Input Definition Fault
	Input Value Fault
	Input Timing Fault
	Input Format Fault
Output	Output Definition Fault
	Output Value Fault
	Output Timing Fault
	Output Format Fault

The software FMEA is performed by applying the failure mode template as in Table 3 to each FBD module. Through the FMEA analysis, some software defects were found even though the program had been tested in a rigorous way. One significant defect is the violation of test criteria. When the channel D is bypassed for a regular manual test, if an operator inadvertently commands the start of a manual test at the channel A, the test-performing processor in the channel A can generate a test start signal without a channel bypass of the channel A. Though this case is actually hard to occur in a real situation (and also the trip-functioning processors have the capability of rejecting this abnormal test start signal based on their own test decision mechanisms), the implemented code contains this defect resulting from a small program logic error that can easily be removed.

REFERENCES

[1] NUREG-0800, Standard Review Plan: BTP HICB–14, Guidance on Software Reviews for Digital Computer-Based Instrumentation and Control Systems, U.S. NRC, 1997.
[2] NUREG/CR-6430, Software Safety Hazard Analysis, Lawrence Livermore National Laboratory, 1995.
[3] G. Y. Park, et al., "Safety Analysis of Safety-Critical Software for Nuclear Digital Protection System," Lecture Notes in Computer Science, Vol.4680, pp.148-161, 2007.
[4] NUREG/CR-6101, Software Reliability and Safety in Nuclear Reactor Protection Systems, Lawrence Livermore National Laboratory, 1993.
[5] N. G. Leveson and T. J. Shimeall, "Safety Verification of Ada Programs using Software Fault Trees", IEEE Software, July, pp.48-59, 1991.
[6] G. Y. Park, et al., Fault Tree Analysis of KNICS RPS Software, Nuclear Engineering and Technology, Vol.40, No.5, pp.397-408, 2008.