Transactions of the Korean Nuclear Society Spring Meeting
Jeju, Korea, May 10-11, 2007

Fault Tree Analysis of KNICS RPS Software

Gee-Yong Park,a Kwang Yong Koh,b Eunkyoung Jee,b Dae Hyung Lee,a Kee-Choon Kwon a
a Korea Atomic Energy Research Institute, 150 Deokjin, Yuseong, Daejeon, 305-353, Korea, gypark@kaeri.re.kr
b Korea Advanced Institute of Science and Technology, 373-1 Guseong, Yuseong, Daejon, 305-701, Korea

1. Introduction

A digital reactor protection system (RPS), being developed in the KNICS (Korea Nuclear Instrumentation & Control System) project, which is called the IDiPS in the KNICS project, contains safety-critical software. The IDiPS is 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 trip signal when a trip condition is satisfied according to the comparison results of at least two channels out of a total of the four bistable channels. All these functions are implemented in the software for the IDiPS, and the trip-functioning software is classified as safety-critical. The safety analysis on the safety-critical software is being performed as a part of the verification and validation (V&V) activities. In this summary, the software safety analysis by a software fault tree analysis (SFTA) is presented.

2. Strategy and Methods

It is recommended in the code and standards that the software safety analysis (SSA) shall be performed during the development of the software used for a safety system of nuclear power plants [1, 2]. In the KNICS project, the software safety analysis is activated at each phase of the software lifecycle. For the techniques for the SSA, the software HAZOP (Hazard and Operability) method is used in the hazard analysis (HA) at the requirements phase and the software HAZOP and SFTA are employed at the design and implementation phases.

The purpose of applying the software HAZOP and SFTA to a software system is to identify a defect or hazard in the software that can induce or affect the system hazard acquired from a system-level hazard analysis by an FMEA (Failure Modes and Effects Analysis) or an FTA (Fault Tree Analysis). For the IDiPS RPS, the software-contributable system hazards were identified through a review of the IDiPS FMEA results and they are presented in Table 1.

Table 1. System 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

The criticality level in Table 1 is given relatively to the severity of a hazard item. The level 4 is the most significant hazard that can drive a plant to a severe accident, and the level 1 indicates an insignificant hazard that seldom affects the system availability. The top node of the SFTA is only for the first hazard item in Table 1 and thus it is confined to the event that a software module cannot generate a trip signal when a trip condition for the software module is satisfied.

In the SSA for the IDiPS software at the design or implementation phase, the software HAZOP was, at first, applied to the software modules represented by a function block diagram (FBD) which is used for the POSAFE-Q, a programmable logic controller (PLC) developed in the KNICS project. The software HAZOP evaluated all the design specifications with respect to all the software-contributable system hazards as in Table 1. And the significant defective areas in the FBD modules were identified by this method. The SFTA was then applied to these defective modules to identify accurately a defective location or a logic error. Both methods are redundant and complementary in that the software HAZOP is a forward (in fact, HAZOP is a bidirectional method but, in this study, a forward analysis was more weighted) and broad-thinking analysis method through a team work of the HAZOP members and, on the contrary, the SFTA is a backward and local systematic analysis method by an individual analyst.

3. Application of Software Fault Tree Analysis

The FTA is a well-established safety analysis technique in nuclear power plants [3] and it has been widely used in the safety analysis. The safety analysis by the FTA in the software is slightly different due to the fact that the software is configured based on the logistic constructs and its behavior is deterministic.

As supposed by Leveson and Shimeall [4], the purposes of the SFTA are to detect software logic errors, to determine the conditions under which fault-tolerance and fail-safe procedures should be initiated, and to facilitate effective safety testing by pinpointing critical functions and test cases. In the SFTA, it is hypothesized that the software has produced an unsafe output and it is shown that this could not happen because the hypothesis leads to a contradiction [4].

The SFTA is usually constructed based on the so-called fault tree templates which are small fault trees for their corresponding components in the software. And one more different aspect of the SFTA is that an event in a fault tree template may be a just logic operation, which is prohibited in the conventional FTA from the fact that all events that are linked together on a fault tree should be written as faults [3]. For a typical function block (FB) in the FBD module, a fault tree template is constructed in a way that failure modes are extracted starting from the output port of an FB, through the body of the FB, ending at the input ports, as shown in Figure 1. The lower left event in Figure 1 indicates plausible faults in an FB and the lower right event is for the logic operation through which a template in the immediate lower tree level is attached.

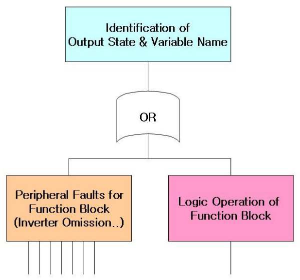

Identification of Output State & Variable Name
OR
Peripheral Faults for Function Block (Inverter Omission..)
Logic Operation of Function Block

Figure 1. Overall architecture for constructing a fault tree template for function blocks in a FBD module.

Figure 2 shows a fault tree template for the AND function block. Based on the templates like that in Figure 2, the SFTA is constructed for the FBD modules selected from the software HAZOP analysis. Figure 3 shows a very simple SFTA among the SFTA results; it depicts the SFTA for a low DNBR trip module in the bistable processor. The FBD module describing a low BNBR trip (DNBR_LO Trip) is shown in Figure 4.

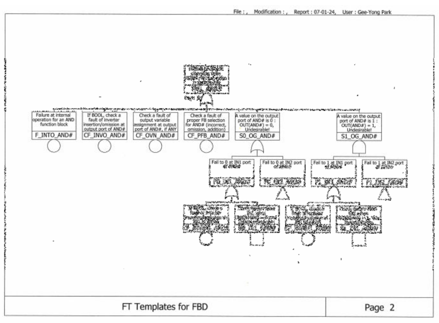
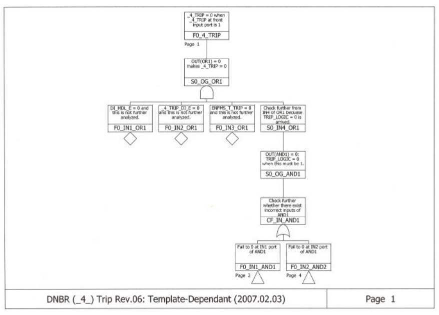
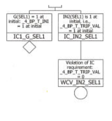


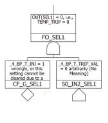
Figure 2. A Fault tree template for the AND function block.



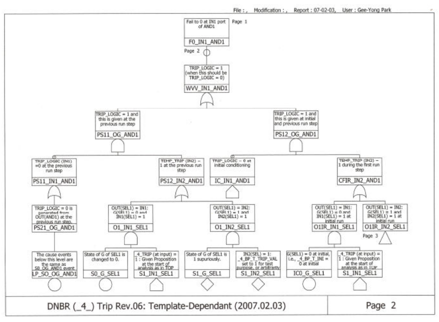
(a)



(b)



(c)



(d)

Figure 3. Software FTA for DNBR_LO trip module.

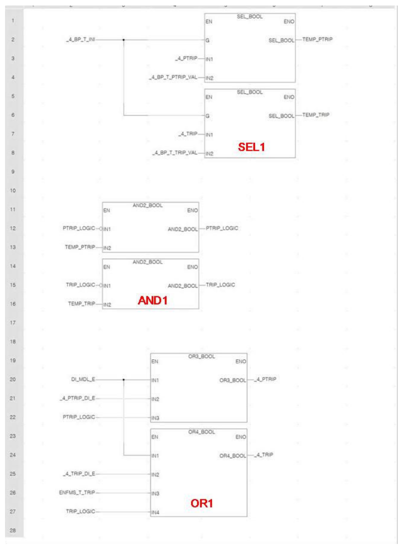


Figure 4. Function block diagram module of BNBR_LO trip.

From Figure 3(d), an internal variable TRIP_LOGIC is toggled between 0 and 1 when a trip condition is satisfied, which means some logic error exists in the AND1 function block. This defect identified by the SFTA had not been identified in the previous processes of a document evaluation and a formal verification.

4. Conclusion

For the SFTA of a digital RPS, its strategy and method are presented in this summary. Because of a different viewpoint from the V&V activities, the SFTA can obtain some valuable results that have not been identified through a rigorous V&V procedure.

REFERENCES

[1] Regulatory Guide 1.168, Verification, Validation, Reviews and Audits for Digital Computer Software Used in Safety Systems of Nuclear Power Plants, U.S. Nuclear Regulatory Commission , 2004.
[2] IEEE Std-1228, Software Safety Plan, 1994.
[3] W. E. Vesely, et al., Fault Tree Handbook, NUREG-C492, U. S. Nuclear Regulatory Commission, 1981.
[4] N. G. Leveson, and T. J. Shimeall, Safety Verification of Ada Programs using Software Fault Trees, IEEE Software, pp. 48-59, July 1991.