A Framework for the Safety Assurance of Safety Software in Nuclear Power Plants

Kee-Choon KWON[1], Jang-Soo LEE[1] and Eunkyoung JEE[2]

1. *Nuclear ICT Research Div., Korea Atomic Energy Research Institute, Daejeon, Rep. of Korea (E-mail: {kckwon, jslee}@kaeri.re.kr)*
2. *School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Rep. of Korea (E-mail: ekjee@se.kaist.ac.kr)*

Abstract: When developing and conducting verification and validation of the safety software in nuclear power plants to receive a license from the regulatory body, it is difficult to judge the safety and dependability of the development, implementation, and validation activities by simply reading and reviewing the documentation. A systematic evaluation technique is necessary to determine whether particular software safety assurance activity defects are at acceptable levels. In this study, we apply a safety case methodology to assess the level and depth of the results of the development and validation performed by the manufacturer to target a bi-stable processor of a digital reactor protection system, and analyze the evaluation results. Also, we assess the hazard analysis techniques to measure the applicability and compare them according to the software development life cycle. We proposed a new framework by applying a modified hazard analysis, including a safety case methodology. We confirmed that it is possible to effectively supplement the existing safety demonstration method.

Keyword: Software Verification and Validation, Software Hazard Analysis, Safety Case, Safety Assurance

1. Introduction

The establishment of reliability and safety is the key point in developing safety-critical software that satisfies the function of a digital reactor protection system in a nuclear power plant. These two features are an important aspect of safety-critical software in the safety instrumentation and control system of a nuclear power plant. Nuclear safety-critical software is under strict regulatory requirements, which are essential for ensuring the safety of nuclear power plants. The verification and validation (V&V) and a hazard analysis of safety-critical software need to follow regulatory requirements through the entire software life cycle. To obtain a license from the regulatory body through the development and validation of safety-critical software, it is necessary to meet the code and standards required by the regulatory body throughout the software development process.[1] Fig. 1 shows the existing safety assurance method for safety critical software in nuclear power plants.

Fig. 1 Safety assurance framework: As-is.

Documents submitted to the regulatory body during the licensing process are vast in number, including software qualification, in other words, V&V, hazard analysis, and configuration management activities. These complex documentation systems and a large amount of review information are not easy to accurately read for the development activities, implementation technology, and validation activities. Therefore, such activities proposed by nuclear power plant manufacturers, in particular, require a systematic evaluation technology that software demonstration

activities can use to determine an acceptable level for software faults. To evaluate the level and depth of the development and validation results, the safety of the reactor protection system established will be guaranteed by applying a safety case methodology.[2]

In the industrial domain, there is an attempt to ensure the safety of a system by developing a system in accordance with the international standards of functional safety, such as IEC 61508[3]. When the safety features developed in accordance with the functional safety standard may increase the safety of the target system, the final product, which is developed according to a well-defined process, cannot guarantee the safety. Safety case technology is attracting attention in that the technology systematically proves and assesses the target system to be safe. When developing a parallel system when creating a safety case, the safety of the system is guaranteed, and can be developed while constantly considering whether and how the system can be safe. If the regulatory body utilizes a safety case methodology, it can be very helpful in assessing the safety of the system systematically.

2. Hazard Analysis

2.1 Hazard Analysis Overview

A hazard analysis is a process that explores and identifies conditions that are not identified by the normal design review and testing process. In order to prevent accidents in which the safety software of the nuclear digital instrumentation and control system malfunctions and leaks radioactivity into the environment, causing damage to property and human life, it is necessary to qualify the safety software with high integrity. Therefore, regulatory bodies and code & standards in each country establish their position on software hazard analysis. We analyze the position of the regulatory bodies and these code & standards, and secure the framework for the safety software hazard analysis by establishing the coverage according to the application of the hazard analysis method. The applicable hazard analysis method according to the software development life cycle is shown in Fig. 2.

A hazard analysis refers to the technique of analyzing the hazards that can cause the failure of the target system to make the best use of expert knowledge, experience, and ability. Although the US NRC did not endorse the content of the hazard analysis contained in Annex D of IEEE Std. 7-4.3.2-2010[4], regulatory bodies did not express a clear stance on a hazard analysis. Recently, however, MDEP (Multinational Design Evaluation Program), digital instrumentation and control working group 10 issued "Common Position on Hazard Identification and Control for Digital Instrumentation and Control Systems." Regulator Task Force on Safety Critical Software has recently started discussing the issue of software hazard analysis, and it has been suggested that a hazard analysis should be adopted.

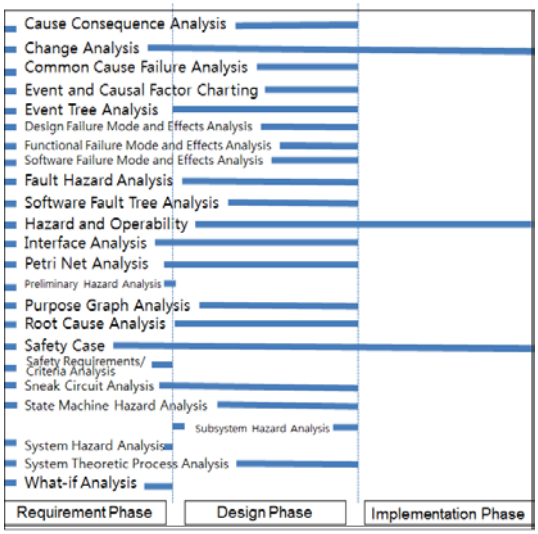

Fig. 2 Hazard analysis based on software development life-cycle

2.1 New Framework for Hazard Analysis

Hazard analysis techniques/tools are methods that help to carry out a hazard analysis effectively. Various methods and supporting tools are used, from traditional methods such as a Fault Tree Analysis (FTA), Hazard and Operability (HAZOP), and Failure Mode and Effects Analysis (FMEA), to new technologies such as System Theoretic Accident Modeling and Process/System Theoretic Process Analysis (STAMP/STPA). The licensing criteria require a hazard analysis of the safety-critical software from each phase of the life cycle. The HAZOP method has been suggested for a hazard analysis in the software requirement and design phases. In the design phase, the software HAZOP was performed first, and a software FTA was then applied. The software FTA was applied to some critical modules selected from the software HAZOP analysis. The software FTA can obtain some valuable results that have not been identified through a rigorous V&V procedure. The hazard analysis for safety-critical software is specifically the differences compared to other non-safety software V&V. In the planning phase, we need to create a safety plan, and conduct a hazard analysis accordingly.[5]

Safety-critical system software continues to perform safety related activities from the development stage. These activities are managed as software safety processes, which are separate procedures, and are carried out in close relationship with software development.

In recent years, new techniques such as STAMP/STPA and safety case have emerged and are

being introduced into practical problem solving. FTA, which is a hazard analysis method in the existing software design phase, was executed only for the suspected part after the execution of HAZOP. However, it is difficult to create a fault tree and it takes a significant amount of time. The performance was small and excluded owing to engineering judgment. Therefore, the proposed new hazard analysis project implementation performs the FMEA at the system requirement phase and the STPA at between the planning phase and the software requirement phase. The purpose of STPA is to derive safe software requirements early in the system development. In the requirements phase, design phase and implementation phase, software HAZOP is performed. In the integration and validation phase, the hazard checklist is used. Fig. 3 shows the As-is and To-be frameworks of a hazard analysis.

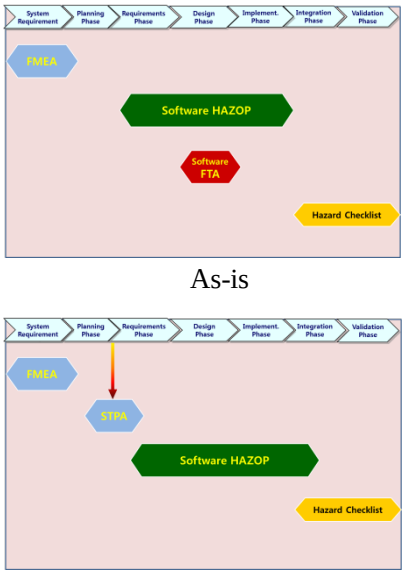

Fig. 3 Hazard analysis framework: As-is and To-be

3. Safety Case

3.1 Introduction to Safety Case

A safety case can be defined as a documented body of evidence that provides a convincing and valid argument in that a specified set of critical claims about a software system's properties are adequately justified for a given application in a given context. A safety case presents an argument in that a software system (e.g., a combination of hardware and software) is acceptably safe, secure, and reliable in a given context. Experience with safety cases has mainly been in the area of safety-critical systems. A safety case requires claims, evidence, and arguments linking evidence to claims[6]:

- Claim: A statement regarding a critical characteristic (i.e., safety, security, reliability) of the system that is being asserted.
- Arguments: Explanations that can be reasonably interpreted as indicating that the critical characteristics are met, usually by demonstrating compliance with the requirements, sufficient mitigation of hazards, or avoidance of hazards.
- Evidence: Results of observing, analyzing, testing, simulating, and estimating the properties of a system that provides fundamental information from which the presence of some system characteristic can be inferred.

A 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.[7] The safety case approach is considered an effective way to argue and evaluate the system safety, and it has been contrasted with prescriptive or process-based approaches, which assume that following the process prescribed in the safety standards will generate evidence for safety.[8] Because both safety argument and prescriptive approaches have their own merits, it is expected that both approaches may complement each other.[2]

Through the study, we used the safety case approach along with the regulatory approach. Thus, during the review process for certification, it was confirmed that there is a possibility of clearer and more efficient communication between the developer and the regulator, focusing on the safety of the target system. Additional effort and expense are required to comply with the code & standards, in addition to applying a safety case. It is therefore necessary to study how to use the regulatory approach and the safety case approach together in an efficient manner.

3.2 Application of Safety Case

An existing safety assurance is applied to build documents in each software life cycle for the software development and validation, and is provided to the regulatory body. Regarding this vast amount of documentation, it is time for the regulatory body or reviewer to review all documents and be convinced that "this software is safe at an acceptable level."

In this paper, existing lists of simple expressions and one-sided enumeration and other methods to ensure the safety presented as part of the Goal Structuring Notation (GSN) technique[9,10], in order to take advantage of the UK's Adelard Assurance and Safety Case Environment (ASCE)[11] version 4.2.7, was used as a software tool. Generally, a safety case is composed of a safety claim, safety evidence, and safety argument for systematic and schematic safety

arguments for systematic and schematic safety assurance.[12] We implemented a safety case for a Bi-stable processor (BP) and Coincidence processor (CP) in a digital reactor protection system. Fig. 4 shows a bird's eye view for a safety case of BP/CP software.

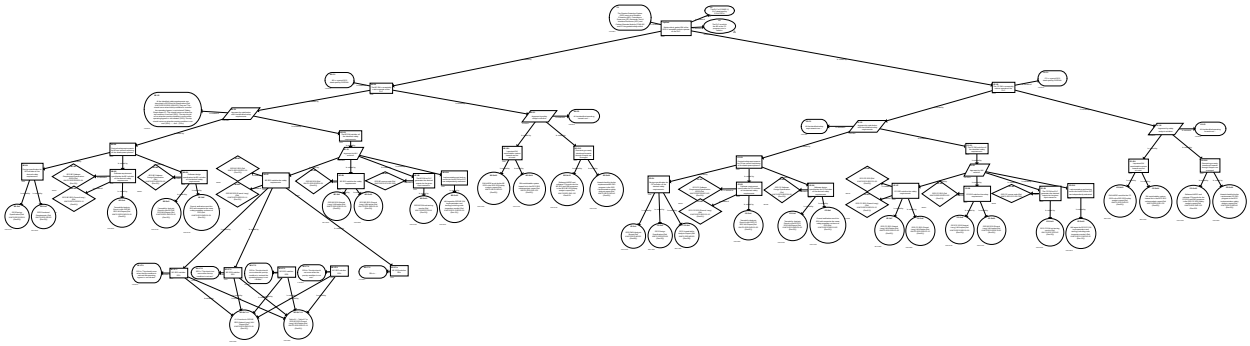

Fig. 4 BP/CP Software safety case: Bird's eye view

The top goal, "Safety-critical graded SW of the RPS is acceptably safe to operate on the PLC," for the BP/CP software, as shown in Fig. 5., C1 states the production of the reactor protection system, and BP is the part of this system.

This top-level goal is divided into BP-G1, "BP software is acceptably safe to perform safety functions in safety grade control devices" and CP-G1, "CP software is acceptably safe to perform safety functions in safety grade control devices". In Figure 5, C1 and C2 are a kind of commentary that is a further explanation for making this claim. C1 describes that the reactor protection system consists of four subsystems and that BP is part of this system, whereas C2 is described by the platform manufacturer and the type of PLC products. A1 is the assumption against the claim that the PLC on which the BP program runs is reliable. In other words, the safety assurance presented here means that it does not consider the influence of the safety grade controller hardware failure.

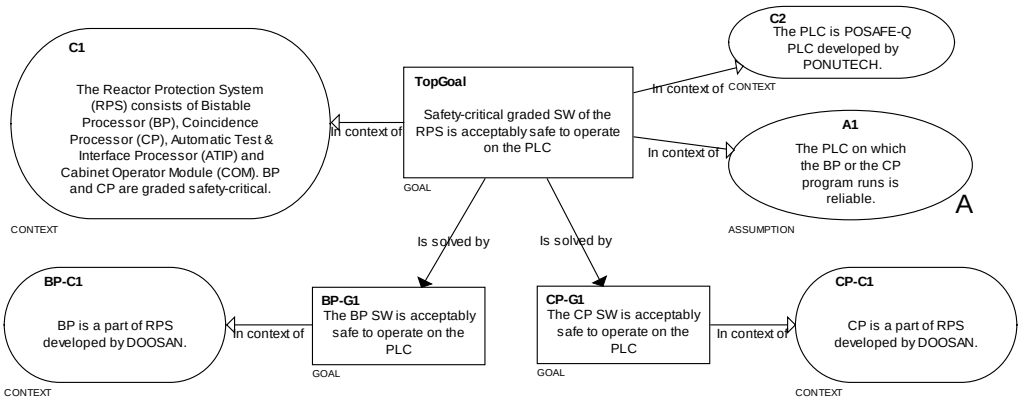

Fig. 5 BP/CP Software safety case: Top goal

BP-G1, as shown in Fig. 6, is the top goal of BP: "The BP software is acceptably safe to operate on PLC." As shown in Fig. 6, to support the claim of BP-G1, two strategies (BP-S1, BP-S2) have been established in this example. One argument is that all required safety requirements are satisfied (BP-S1), and the other is an argument (BP-S2) that demonstrates the safety through a hazard analysis. Fig. 6 shows the strategy in which BP-S1 is solved by BP-G2 and BP-G3. It can be claimed that the BP software is acceptably safe to operate on a PLC if the desired safety requirements for BP are not missed during all development phases (BP-G2), and BP software satisfies all identified safety requirements (BP-G3). BP-S2 is solved by BP-G11, that is, "Important SW contributable system hazards are not missed," and BP-G12, that is, "Remaining or newly introduced hazards through lifecycle are managed.

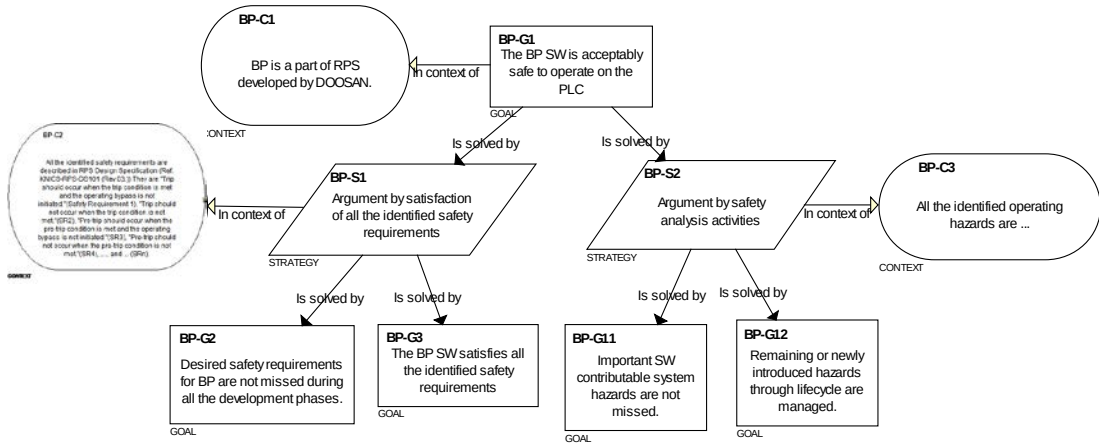

Fig. 6 BP Software safety case: BP-G1

As shown in Fig. 7, the BP-G2 goal claiming that "the desired safety requirements for BP are not missed during all the development phases" can be split into three sub goals: "the design specification for BP includes all the desired safety requirements" (BP-G4), "the software requirement specification for BP includes all the desired safety requirements" (BP-G5), and "the software design specification for BP includes all the desired safety requirements" (BP-G6).

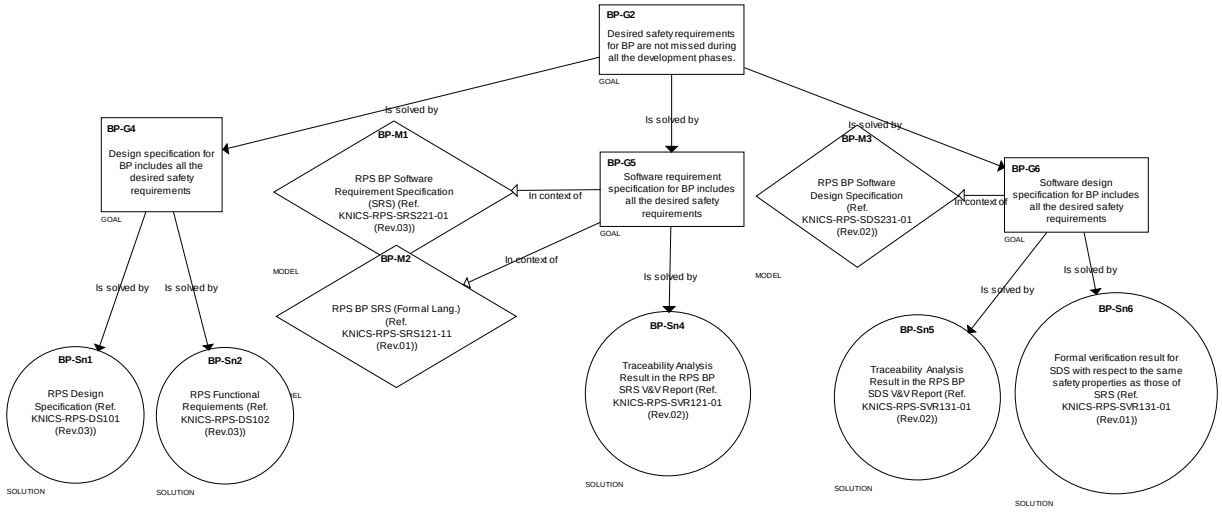

Fig. 7 BP Software safety case: BP-G2

Fig. 8 shows 3 methods demonstrating BP-G3, the claim that "BP software requirements specification satisfies the safety requirements" (BP-G7), and "BP software design specifications satisfy higher safety requirements" (BP-G8). A safety claim is that "the BP SW on PLC generates the desired outputs for the given input scenarios" (BP-G9) and the claim, "implementation and testing results for the BP SW on PLC are independently evaluated" (BP-G10).

BP-G7 in Fig. 9 is the claim, "BP SRS should meet the safety requirement," and originally the evidence (natural language SRS V&V report, and formal SRS V&V report) directly connected to the below. We modified this part as the BP-G7 split into n (number of safety requirement) sub-goals. BP-G7.1 to BP-G7.n, "the BP SRS satisfies the safety requirement n." Evidence was also the entire report of the RPS BP SRS V&V, which is a little more specific, as in BP-Sn7.1.1 at the bottom of the figure below, in Section 6.1.5 of the RPS BP SRS V&V Report. We represented this in a slightly more specific way as "Table 6.1 through Table 6.7 in the formal SRS V&V report," in BP-Sn7.1.2.

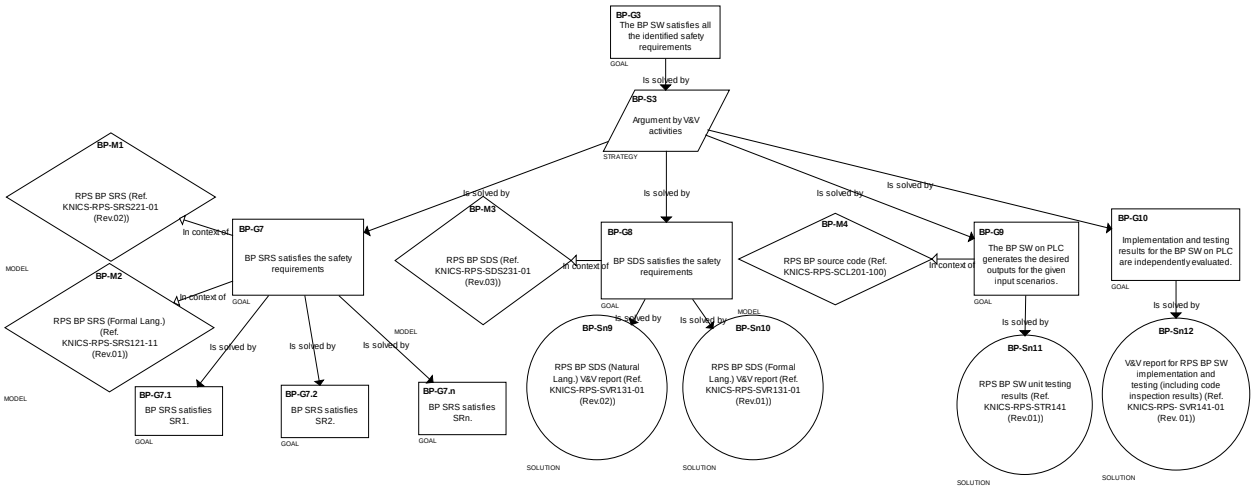

Fig. 8 BP Software safety case: BP-G3

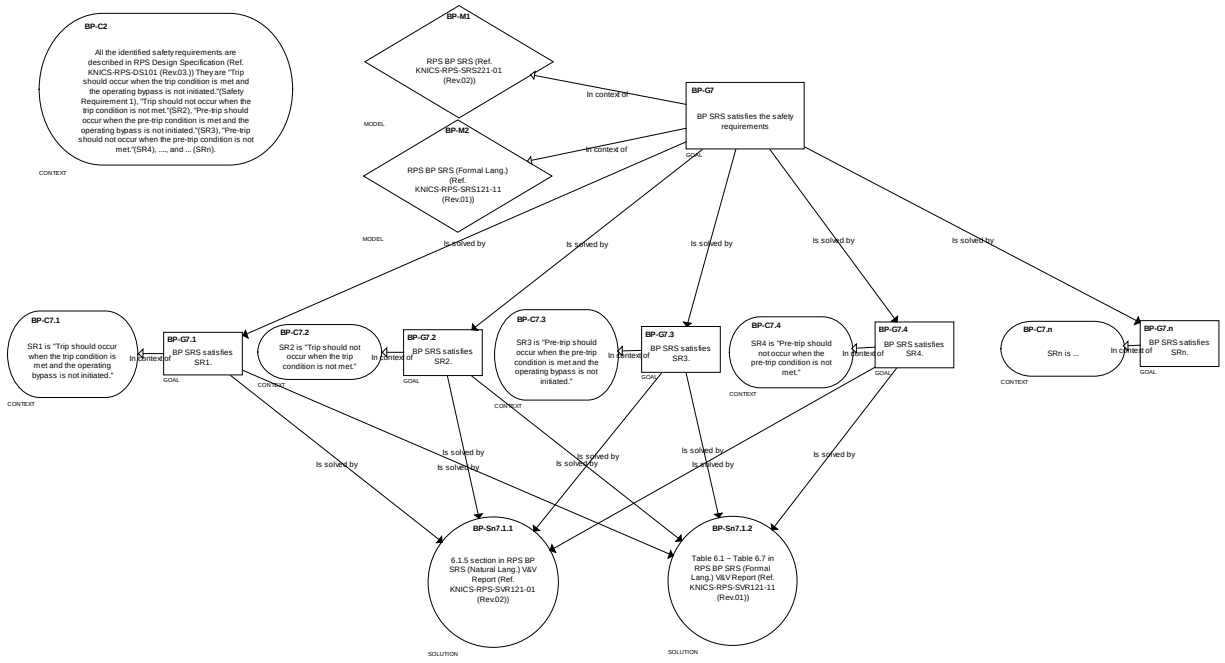

Fig. 9 BP Software safety case: BP-G7

Fig. 10 describes how the top claim (the safety of the BP SW) can be argued based on the hazard analysis activities, in addition to the satisfaction of the safety requirements. As illustrated in Fig. 10, if important SW contributable system hazards are not missed (G11) and the remaining or newly introduced hazards through the lifecycle are managed (G12), the BP SW can be claimed to be acceptably safe to operate on the PLC. The software HAZOP was performed in the software hazard analysis during the requirements phase of the BP development, and software HAZOP and software FTA techniques were used in the design and implementation phase. Thus, software HAZOP results for the BP SRS in the RPS SRS hazard analysis report (BP-Sn13) and the software contributable system hazard list in the RPS SDS hazard analysis report (BP-Sn14) can both serve as evidence supporting claim BP-G11. BP-Sn15 and BP-Sn16 exist on the basis of BP-G12. BP-Sn15 is the "Software HAZOP and software FTA results for the RPS SDS and FBD programs in the safety analysis report," and BP-Sn16 is a "Hazard checklist for the implemented BP FBD programs in the RPS implementation safety analysis report."

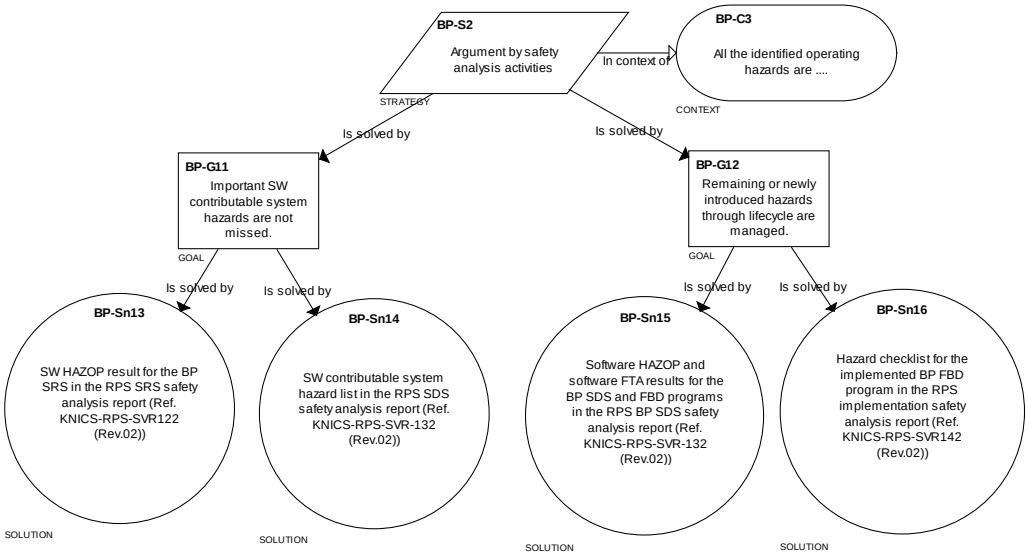

Fig. 10 BP Software safety case: Hazard analysis

3.3 Analysis and Evaluation

Through a case study, applying the safety case approach for the target system, which was developed and qualified through a prescriptive methodology, we confirmed the advantages and disadvantages as follows by using a prescriptive and safety case approach.

First, using the safety case approach, we confirmed the possibility of clearer and more effective communication between the developer and the regulator during the certification review process, focusing on the safety of the target system. In the case of the reactor protection system, hundreds of documents were produced and submitted to the regulatory body, which took considerable effort and time to review. The relevance of each output to system safety is different and it is not easy to determine which parts of the output are more important and less important in terms of system safety. Even if developed in accordance with the prescriptive approach, safety-focused and effective communication will be possible if the outputs are expressed using a safety case and submitted to the regulatory body.

On the other hand, additional effort and expense are required to comply with safety software standards, and in addition, to prepare a safety case. There are studies on safety case patterns, but much of the safety case creation and management still depend on manual work. We must continue our efforts to develop appropriate guidelines and tools to create and manage the safety case, and we need to study how to use regulatory and safety case approaches in an efficient manner.

It was also found that using a safety case method was not enough to show that all the requirements of the system were met. Requirements may include safety requirements as well as other aspects such as security and performance. The safety case focuses on the safety aspects and therefore does not address all of the other requirements. It may be difficult to replace the prescriptive approach with the use of the safety case method when developing and complying with many international standards related to safety as well as other quality attributes.

4. New Framework for Safety Assurance

Based on the summary of the positions of regulatory bodies and code & standards for hazard analysis of nuclear safety software, a hazard analysis method to be applied to each period of safety software life cycle of nuclear power plant was derived. A new framework applying a safety case to a framework that achieves safety assurance through software V&V and a hazard analysis by applying safety cases based on verification logic of digital reactor protection systems performed by concurrent logical processors to provide systematic assessment techniques to determine if software defects are acceptable. We have reviewed the results of the safety software hazard analysis, the classification of applicable hazard analysis method according to software development procedures, and the development of hazard analysis template for evaluation.

We applied the safety case as an argument, which is the upper objective that the system should satisfy, the argument that connects the claim and evidence, the confirmation and verification, and the result of the hazard analysis as evidence supporting the claim. We

proposed a new framework for safety assurance (To-be) by combining upgraded hazard analysis framework and safety case and by modifying the framework (As-is) introduced in the introduction, as shown in Fig. 11.

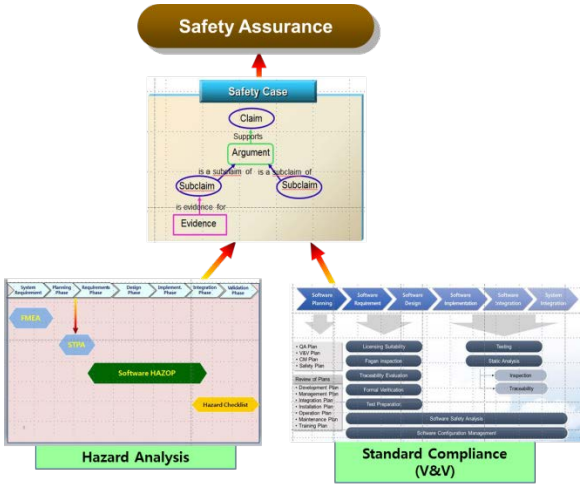

Fig. 11 Safety assurance framework: To-be

5. Conclusions

Based on the summary of the trends of regulatory bodies and code & standards for a hazard analysis of nuclear safety software, a hazard analysis method to be applied to each period of safety software life cycle of a nuclear power plant was derived, along with a hazard analysis methodology coverage and hazard analysis method template. We proposed a new framework to remove the design phase FTA from the existing framework and to perform the STPA in the middle of the planning phase and the software requirement phase in the safety software hazard analysis applied to the nuclear safety system of nuclear power plant. The purpose of STPA is to derive safe software requirements early in the system development stage.

We introduced a new framework by applying a safety case to the framework that achieves safety assurance by software verification and validation and hazard analysis, based on the qualification of the digital reactor protection system bi-stable logic processor that has been performed. We provided systematic evaluation techniques to determine if software defects are acceptable. We applied the safety case as an argument, which is the upper objective that the system should satisfy, the argument that connects the claim and evidence, and the result of verification and validation and the hazard analysis as evidence supporting the claim.

When using the Adelard tool by applying a safety case method, there is a possibility that different results may be derived depending on the engineer to write, and thus we developed safety case writing and evaluation guidelines. As a result of this study, it is necessary to develop a method for determining whether the above arguments are sufficient, and a technique for evaluating the adequacy of the evidence linking the presented claims and evidence. It is necessary to study how to use the regulatory approach and the safety statement approach together in an efficient manner.

Acknowledgement

This work was supported by the Nuclear Safety Research Program through the Korea Foundation of Nuclear Safety (KOFONS), granted financial resource from the Nuclear Safety and Security Commission (NSSC), Republic of Korea (No.1403008), and by a grant from the Korea Ministry of Science and ICT, under the development of the framework of I&C dependability assessment.

References

[1] K. Kwon, J. Lee, G. Park, E. Jee, “Paradigm of Software Safety Assurance for Digital Reactor Protection System in NPPs,” Proceedings of the KCC2016, Jeju, Korea, June 29-31, 2016. (In Korean)

[2] E. Jee, G. Park, J. Lee, K. Kwon, “A Safety Case for Reactor Protection System Software Developed with a Prescriptive Approach,” EHPG 2016, Oslo, Norway, May 10, 2016.

[3] IEC 61508, Functional Safety of Electrical/Electronic/Programmable Electronic Safety Related Systems, 2010.

[4] IEEE Std. 7-4.3.2, IEEE Std. Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations, 2010.

[5] K. Kwon, et al., Qualification of safety-critical software for digital reactor safety system in nuclear power plants, Nuclear Safety and Simulation, Vol.4, No.3, pp. 226-233, 2013.

[6] IEEE Std. 730, “IEEE Standard for Software Quality Assurance Processes,” 2014.

[7] MoD, Defence Standard 00-56 Issue 4 (Part 1): Safety Management Requirements for Defence Systems, UK Ministry of Defence (MoD).

[8] R. Hawkins, I. Habli, T. Kelly and J. McDermid, "Assurance cases and prescriptive software safety certification: A comparative study," Safety Science, vol. 59, p. 55–71, 2013.

[9] Tim P. Kelly, “Arguing safety: A Systematic Approach to Managing Safety Cases,” University of York, 1999.

[10] John Spriggs, GSN – The Goal Structuring Notation : A Structured Approach to Presenting Arguments, Springer, 2012.

[11] https://www.adelard.com/.

[12] Terje Siversten, Software Safety Demonstration, Halden Reactor Project, HWR-1056, 2013.