Transactions of the Korean Nuclear Society Autumn Meeting
Busan, Korea, October 27-28, 2005

Software safety Analysis on the Model Specified by NuSCR and SMV Input Language at Requirements Phase of Software Development Life Cycle using SMV

Kwang Yong Koh, Poong Hyun Seong
Department of Nuclear and Quantum Engineering, Korea Advanced Institute of Science and Technology, 371-1, Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
goeric1@kaist.ac.kr, phseong@kaist.ac.kr

1. Introduction

Safety-critical software process is composed of development process, verification and validation (V&V) process and safety analysis process. Safety analysis process has been often treated as an additional process and not found in a conventional software process. But software safety analysis (SSA) is required if software is applied to a safety system, and the SSA shall be performed independently for the safety software through software development life cycle (SDLC) [1]. Of all the phases in software development, requirements engineering is generally considered to play the most critical role in determining the overall software quality. NASA data demonstrate that nearly 75% of failures found in operational software were caused by errors in the requirements. The verification process in requirements phase checks the correctness of software requirements specification, and the safety analysis process analyzes the safety-related properties in detail [2], [3].

In this paper, the method for safety analysis at requirements phase of software development life cycle using symbolic model verifier (SMV) [4] is proposed. Hazard is discovered by hazard analysis and in other to use SMV for the safety analysis, the safety-related properties are expressed by computation tree logic (CTL) [5].

2. Safety analysis at requirements phase using SMV

In this section, scope, process and method for this research and basic theory or knowledge necessary for the proposed method are described.

2.1 Safety Analysis

Safety process generally comprises of four stages and each of them has its own main task [6]. Among them, only hazard analysis is performed in this research. That is because safety requirements and designation of safety-critical systems stages are strongly related to software development, and my research concern is not the software development itself. And also safety validation is substituted to safety verification because the application model is a small part of the system, not whole system. Table 1 illustrates safety processes and their main tasks.

Table 1. Safety process and main task.

Safety Analysis Process	Main Task
Hazard and risk analysis	Assessing the hazards and the risks of damage associated with the system
Safety requirements specification	Specifying a set of safety requirements which apply to the system
Designation of safety-critical systems	Identifying the sub-systems whose incorrect operation may compromise system safety
Safety validation	Checking the overall system safety

2.2 Hazard Analysis

Hazard analysis is 1) to identify all possible hazards potentially created by a product, process or application and 2) structured into various classes of hazard analysis and carried out throughout software process. And 3) a risk analysis should be carried out and documented for each identified hazard [6]. But in this work, it is limited only at requirements phase and the risk analysis is not performed because only qualitative analysis is carried out and quantitative analysis is not included. Therefore, although there are four stages in hazard analysis, this research covers only hazard identification which is core work of the hazard analysis. Table 2 shows hazard analysis stages and their main tasks.

Table 2. Hazard analysis stages and main task.

Hazard Analysis Stages	Main Task
Hazard identification	Identifying potential hazards which may arise
Hazard classification	Assessing the risk associated with each hazard
Hazard decomposition	Decomposing hazards to discover their potential root causes
Safety specification	Defining how each hazard must be taken into account when the system is designed

2.3 Computation Tree Logic (CTL)

CTL is a kind of temporal logics used by model checking tools and serves to formally state properties concerned with the executions of a system [5]. There are several categories of system properties expressed by CTL, but only reachability and safety property which are directly related to system safety are matters of concern in this research.

If a system can't reach a hazard, we can say that the system is safe from the hazard. Therefore, by investigating non-reachability of a hazard, a system or model of system can be checked with respect to its safety against the hazard. In fact, reachability and safety properties have exactly opposite meaning [5]. That is, non-reachability and safety property have exactly same meaning. Table 3 shows definitions and CTL expression of reachability and safety property, and relationship between them in CTL expression.

Table 3. Definition and CTL expression of reachability and safety property and relationship between them.

	Reachability Property	**Safety Property**
Definition	Some particular situation can be reached	Under certain conditions, an event never occurs
CTL expression	EF P	AG !P
Relationship	!(EF P) ≡ AG !P	

2.4 Symbolic Model Verifier (SMV)

The SMV system is a tool for checking finite state systems against specifications in the temporal logic CTL [4]. It is usually used for formal verification, although the way to use it for safety analysis is not much different from that to use it for formal verification, but it is used for safety analysis at requirements phase in this work.

2.5 Safety Analysis using SMV

The process of the proposed method for safety analysis is as follows.

1) By hazard analysis, identify potential hazards from a model. 2) Track possible paths leading to hazard using backward search technique which starts with a final event or state and determines the preceding events or states. 3) With CTL operators, translate the paths into CTL expression. 4) Check non-reachability of hazard using SMV. Here, non-reachability and safety property have exactly same meaning in CTL expression as explained above and SMV is used in the same way as used for verification. 5) Finally, after checking non-reachability of hazard using SMV, we can judge whether a model is safe or unsafe from a particular hazard with verification result. Figure 1 illustrates the schematic process of safety analysis at requirements phase using SMV

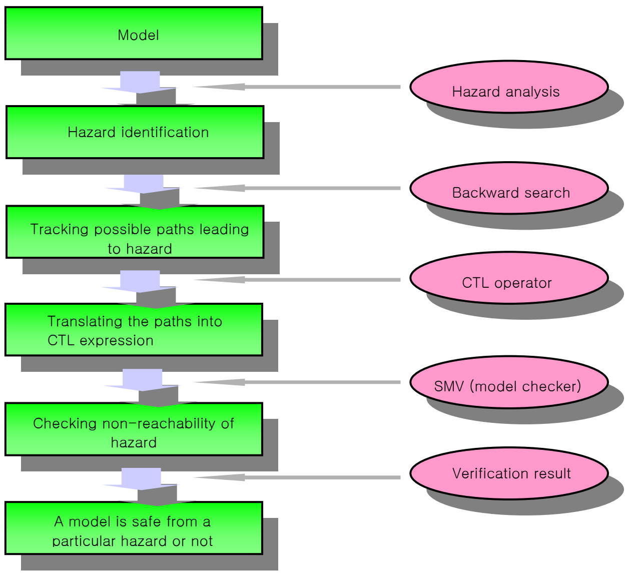
Model
Hazard analysis
Hazard identification
Backward search
Tracking possible paths leading to hazard
CTL operator
Translating the paths into CTL expression
SMV (model checker)
Checking non-reachability of hazard
Verification result
A model is safe from a particular hazard or not

Figure 1. Schematic process of safety analysis at requirements phase using SMV.

3. Conclusion

Because there is no such thing as absolute safety, when mentioning safety, we can say that a system or model of system is safe from only this or that hazard, that is, a particular hazard [6]. In this work, the method for analyzing whether a system or model is safe from a particular hazard by checking non-reachability of hazard using SMV is proposed. Although this research doesn't cover whole safety analysis and safety analysis is simplified, the proposed method is convenient and useful for analyzing system safety against a specific hazard.

REFERENCES

[1] Kyung H. Cha. et al., "Techniques and Tool for Software Qualification in KNICS," Proceeding of the KNS 04 Autumn Conference, Yongpyong, Korea, pp. 591-592 Oct. 28-29, 2004.
[2] R. Lutz, "Targeting Safety-Related Errors during Software Requirements Analysis," The Journal of Systems and Software, vol.34, pp. 223-230, Sept. 1996.
[3]Taeho Kim, "Property-based Theorem Proving and Template-based Fault Tree Analysis of NuSCR Requirements Specification," Doctorial Thesis, Department of Electrical Engineering & Computer Science, Division of Computer Science, KAIST, Dec. 2004.
[4] K. L. McMillan, "The SMV system for SMV version 2. 5. 4", Nov. 6, 2000.
[5] B. Berard, M. Bidoit, et al., "System and Software Verification: Model-Checking Techniques and Tools," Springer, 2001.
[6] Nancy G. Leveson, "SAFEWARE – System Safety and Computers," Addison-Wesley Publication Company, New York, 1995.