Proceedings of the Korean Nuclear Society Spring Meeting

Cheju, Korea, May 2001

Development of Software Safety Analysis Method for Nuclear Power Plant I&C Systems in Requirement Specification Based on Statechart and SCR

Jung Hwan Lee, Seo Ryong Koo, Han Seong Son, Poong Hyun Seong

Korea Advanced Institute of Science and Technology

373-1 Kusong-Dong, Yusong-Gu

Taejon Korea 305-701

Abstract

In recent years, Instrumentation and Control (I&C) system based on digital computer technology has been widely used throughout industries. These industries such as Nuclear Power Plant (NPP) have safety critical systems. Thus, safety critical system must have sufficient quality to assure a safe and reliable design. In this work, a formal requirement analysis method for Nuclear Power Plant (NPP) Instrumentation and Control (I&C) systems is proposed. This method use the Statechart diagram, Software Cost Reduction (SCR) formalism and ISO table newly suggested in this paper for checking the modeled systems formally. The combined method of utilizing Statechart, SCR and ISO table has the advantage of checking the system easily, visually and formally. This method is applied to the Water Level Monitoring System (WLMS). As a result of the formal check, one reachability error is detected.

I. Introduction

In recent years, Instrumentation and Control (I&C) system based on digital computer technology has been widely used throughout industries. These industries such as Nuclear Power Plant (NPP) have safety critical systems. If these safety critical systems had serious failures, the consequences would have an effect on the public health and safety. Thus, safety critical system must have sufficient quality to assure a safe and reliable design. For that reason, the safety and reliability of system must be considered when it is developed. If not, it would spend plenty of time, efforts and cost.

Up to the present, there have been many analysis methods proposed for the safety verification. However, in this paper, a new method for safety verification is introduced. This method is to develop how

to verify the safety of system in Requirement Specification Based on Statechart and Software Cost Reduction (SCR) with several checklists.

The Statechart [1] is an extended form of conventional finite state machines. It provides the concepts of hierarchical and event broadcasting. These features make Statechart a very powerful language to specify complex systems. However, it supports only simulation-based checking of safety properties, that is, verification is performed heuristically and no decisive conclusion can be made. [2] Formal specification can lessen requirements errors by reducing ambiguity and imprecision and by making instances of inconsistency and incompleteness obvious. Given a formal requirements specification, formal analysis can detect many classes of errors, some automatically. Among formal specification, SCR is selected in this work since tables are easy to understand and the tabular notation facilitates industrial application of the SCR method. [3]

The procedure for formal requirement analysis is composed of system modeling and formal checking. System is visually modeled using Statechart, and then it is checked with SCR that is converted from Statechart and Input-State-Output (ISO) table that is newly proposed in this paper.

The main focus of this work lies in the development of a requirement analysis method through combining Statechart, SCR and ISO table which is developed newly in this paper.

This paper is organized as follows. In chapter II, we explain the Statechart and SCR used in this paper for formal requirement verification. In chapter III, we introduce a formal requirement analysis method using several checklists. Water Level Monitoring System (WLMS) was used as an example to illustrate the method introduced in this paper, in chapter IV. A conclusion of the paper and further work is included in chapter V.

II. Related Works: Statechart and SCR

II.1 Statechart

The Statechart proposed by D. Harel expand upon the State machine by providing constructs for hierarchy (the decomposition of states), concurrency (being in more than one state at a time) and broadcasting. (the communication between concurrent states) [1][4] Hierarchy in Statecharts may be looked at from two perspectives. From a top-down viewpoint, hierarchy is the decomposition of a state into more basic states. From bottom-up, hierarchy provokes the ability to encapsulate basic states that have a common behavior with a higher level state. A state may be decomposed into concurrent (orthogonal) sub-states. The encompassing state is designated an and-state and the construct to decompose the state is called an and-line. A state may be decomposed to any number of concurrent components using the and-line construct. This construct denotes the notion of concurrent sub-states. This

notion of being in more than one state at any given time is handled through the principle of concurrency (orthogonality). The combination of hierarchy and concurrency simplify the modeling of large, complex systems by extending the principles of state transition diagrams and reducing the clutter associated with them. Concurrent components (or individual Statecharts) rarely exist independently. It is therefore, necessary to synchronize their interaction. This is accomplished through the principle of broadcast. [1]

Movement from one state to another state is represented by a transition arrow drawn from the source state to the target state. A textual label is placed on this transition to indicate the stimulus and/or the condition under which the transition is to occur and what, if any, resultant action is to take place. The syntax of a label is:

event [condition] / action

An event is the instantaneous occurrence of a stimulus (trigger), a condition is the presence or absence of a value over time(for example, Boolean true or false) and an action is the event or manipulation which occurs when transitioning from one state to another. Figure II.1 is example of Statechart diagram.

II.2 Software Cost Reduction (SCR)

A recent study of industrial application of formal methods concludes that formal methods, including those for specifying and analyzing requirements, are “beginning to be used seriously and successfully by industry . .to develop systems of significant scale and importance” [5]. Included in the study is the SCR method for specifying requirements. Introduced more than a decade ago to describe the functional requirements of software unambiguously and concisely, the SCR method has been extended recently to describe system, not just software or requirements and to incorporate techniques for representing nonfunctional requirements, such as timing and precision. Formal specification can lessen requirements errors by reducing ambiguity and imprecision and by making instances of inconsistency and incompleteness obvious. Given a formal requirements specification, formal analysis can detect many classes of errors, some automatically.

Among formal specifications, SCR is selected in this work since tables are easy to understand and the tabular notation facilitates industrial application of the SCR method. The tables in SCR specifications consist of condition tables, event tables, and mode transition tables. Each table defines a function. A condition table describes an output variable or a term as a function of a mode and a condition. An event table describes either as a function of a mode or an event. A mode transition table describes a mode as a function of another mode and an event. While condition tables define total functions, event tables and mode transition tables may define partial function because some events cannot occur when certain conditions are true [3]. Figure II.2, II.3, II.4 are typical forms of condition table, event table, mode transition table, respectively.

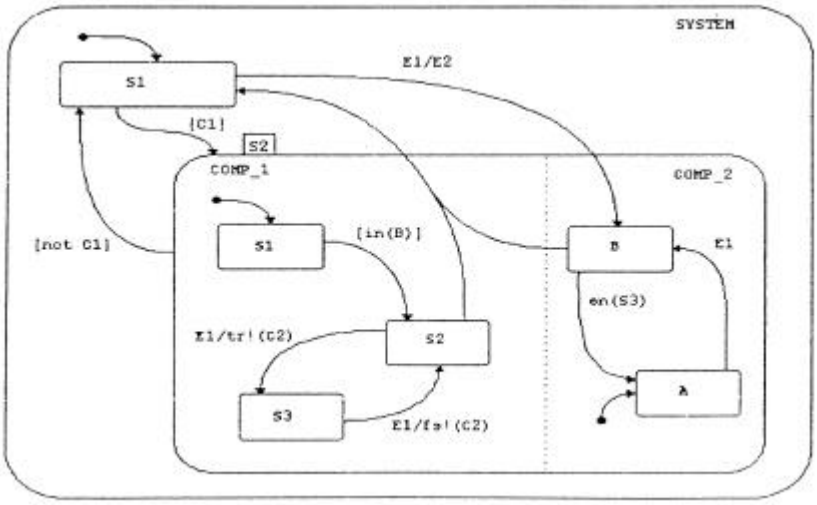

Fig II.1 Example of Statechart diagram

Modes	Conditions			
m_1	$c_{1,1}$	$c_{1,2}$	…	$c_{1,p}$
m_2	$c_{2,1}$	$c_{2,2}$	…	$c_{2,p}$
…	…	…	…	…
m_n	$c_{n,1}$	$c_{n,2}$	…	$c_{n,p}$
r_i	v_1	v_2	…	v_p

Fig II.2 Condition Table

Modes	Events			
m_1	$e_{1,1}$	$e_{1,2}$	…	$e_{1,p}$
m_2	$e_{2,1}$	$e_{2,2}$	…	$e_{2,p}$
…	…	…	…	…
m_n	$e_{n,1}$	$e_{n,2}$	…	$e_{n,p}$
r_i	v_1	v_2	…	v_p

Fig II.3 Event Table

Old Mode	Events	New Mode
m_1	$e_{1,1}$	$m_{1,1}$
	$e_{1,2}$	$m_{1,2}$

	… $e_{1,k1}$	… $m_{1,k1}$
…	…	…
m_n	$e_{n,1}$ $e_{n,2}$ …` $e_{n,kn}$	$m_{n,1}$ $m_{n,2}$ … $m_{n,kn}$

Fig II.4 Mode Transition Table

III. Formal Requirement Analysis Method

III.1 Formal method

In developing of safety critical system such as Nuclear Power Plant, it must be considered safety, reliability and effectiveness of system. Generally, development of software is generally composed of 5 phases, which are requirement analysis, design, implementation, testing and maintenance. Among these phases, requirement analysis phase is the most important in order to assure a safe and reliable design. Thus, formal method is used so as to perform the safety analysis and to examine the consistency and completeness in requirement analysis phase.

Formal methods are approaches, based on the use of mathematical techniques and notations, for describing and analyzing properties of software systems. That is, descriptions of the system are done using notations which are based on mathematical expressions rather than a natural language.

Formal methods offer two principles benefits in software development. First the use of a precise mathematical notation eliminates the inaccuracy and ambiguity that is inherent in natural language expressions. Formal methods also make it possible to analyze or prove, via rigorous mathematical techniques, certain properties of software descriptions. A developer may prove that a specification is complete and internally consistent or that the products of given phase of the software process are consistent with those of a previous phase.

In this work, the formal requirement analysis method involves the following steps. First, the system requirements are modeled using Statechart. This model offers an advantage to the verification process in that the easy communication between use and developer is possible. Then, this is converted to SCR specification and is checked using selected checklists and ISO table.

III.2 Visual modeling using Statechart

Visual modeling is a way of thinking about problems using models that depict real-world ideas in the visual method. Therefore, models are useful for understanding problems and communicating with everyone involved in the project (customer, domain, experts, analysts and designers). Modeling promotes better understanding of requirements, cleaner designs and more maintainable systems. In this work,

system requirements are modeled using Statechart diagrams.

III.3 Converting Statechart to SCR Specification

For the formal check of the system requirements, SCR is selected in this work because tables are easy to understand and the tabular notation facilitates industrial application of the SCR method. The tables in SCR specifications are condition tables, event tables, and mode transition tables. Each table defines a function. A condition table describes an output variable or a term as a function of a mode and a condition, an event table describes either as a function of a mode and an event. A mode transition table describes a mode as a function of another mode and an event.

In order to convert Statechart diagrams to SCR specifications easily, the converting procedure is defined in this work. The first step of the procedure is to find an initial mode in Statechart diagrams and start SCR specification from this initial mode. The second step is to define each operation of class as each mode of SCR specification. The third step is to assign the location of modes using the numbered events in Statechart diagrams and then to convert the states to modes and the trigger events to events in sub diagrams. The fourth step is to convert each diagram to each SCR specification. Finally, in final state, we convert [trigger event / output event] to condition table in each page because condition table represents the total function of the system.

III.4 Safety checklists

The approach presented in Jaffe et. al. (1991) work [6] is to build a formal, finite-state model of the requirement specifications and then to analyze this model to ensure that its properties match the desired behavior (e.g. determinism). The authors accomplish this by stating criteria (usually formal predicates). In this section, however, the safety checklist was developed as a translation of the criteria into an informal, natural language format. These checklists are represented in [7] and we use them partially no.4 determinism, no.14 reachability and no.15 hazard analysis. And they are added coverage, disjointness for consistency check and completeness. But last checklist is not applied in this paper. Although this checklist is the most important of these checklists, thus it is performed in future work.

III.5 Safety verification using selected checklists

In this work, SCR specification converted from Statechart are checked formally for finding errors and ambiguities of system requirements. The checklists used in this paper are enumerated below. These checklists, which determine whether the specifications are well formed, are independent of particular system state. They are the forms of static analysis. [3]

Coverage: Each condition table satisfies the Coverage property. That is, each variable described by a condition table is defined everywhere in its domain

Disjointness: To make the specifications deterministic, each condition, event, and mode transition table must satisfy the disjointness property. That is, in a given state, each controlled variable and term has a unique value, and if a state transition occurs, the new state is unique.

Determinism: On a given input, the systems always follow the same path through the code ,that is, system' s behavior is deterministic.

Completeness: Each variable and modes in mode transition tables is defined.

Reachability: All modes and modules of the specified software are reachable. (used in some path through the code)

The checklists , , , are used with SCR tables composed of condition table, event table and mode transition table. These are checked by making use of following method. [3]

Each SCR table describes a table function, called F_i, for some entity r_i. Table functions define the values of the output variables, terms, and mode classes in SCR requirements specifications. Each entity defined by a table is associated with exactly one mode class, M_j, $1 \le j \le N$. To represent the relation between an entity and a mode class, we define a function μ, where $\mu(i) = j$ if entity r_i is associated with mode class M_j. Using this notation, $M_{\mu(i)}$ denotes the mode class associated with entity r_i.

Figure II.2, 3, 4 are the typical formats of condition, event and mode transition table, a representation of the information in each table as a relation ρ_i, and a set of properties which guarantee that ρ_i is a function. Given ρ_i, we can derive the table function F_i. Figure II.2 shows a typical format for a condition table with n+1 rows and p+1 columns. Each condition table describes an output variable or term r_i as a relation ρ_i between modes, conditions, and values, i.e., $\rho_i = \{(m_j, c_{j,k}, v_k) \in M_{\mu(i)} X C_i X TY(r_i)\}$, where C_i is a set of conditions defined on entities in RF that is a set of entity names r. ρ_i has the following four properties:

1. The m_j and the v_k are unique.

2. $\mathbf{Y}_{j=1}^{n} mj = M_{\mu(i)}$ (All modes are included).

3. For all j: $\bigvee_{k=1}^{p} c_{j,k} =$ true (Coverage check: The disjunction of the conditions in each row of the table is true.)

4. For all j, k, l, $k \neq l$: $c_{j,k} \wedge c_{j,l} =$ false (Disjointness check: The conjunction of the conditions in each row of the table is false).

These properties guarantee that ρ_i is a function.

Figure II.3 illustrates a typical format for an event table with n+1 rows and p+1 columns. Each event table describes an output variable or term r_i as a relation ρ_i between modes, conditioned events, and values, i.e., $\rho_i = \{(m_j, e_{j,k}, v_k) \in M_{\mu(i)} X E_i X TY(r_i)\}$, where E_i is a set of conditioned events defined on entities in RF. ρ_i Has the following two properties:

1. The m_j and the v_k are unique.
2. For all j, k, l, $k \neq l$: $e_{j,k} \wedge e_{j,l}$ = false (Determinism check: The conjunction of the conditioned events in each row of the table is false).

These properties and assumptions on input events guarantee that ρ_i is a function.

Figure II.4 shows a typical format for a mode transition table. A mode transition describes an entity r_i that names a mode class $M_{\mu(i)}$. The table describes ri as a relation ρ_i between modes, conditioned events, and modes, i.e., $\rho_i = \{(m_j, e_{j,k}, m_{j,k}) \in M_{\mu(i)} X E_i X M_{\mu(i)}\}$, where E_i is a set of conditioned events defined on entities in RF. ρ_i has the following four properties:

1. The m_j are unique.
2. For all $k \neq k'$, $m_{j,k} \neq m_{j,k'}$, and for all j and for all k, $m_j \neq m_{j,k}$.
3. For all j, k, k', $k \neq k'$: $e_{j,k} \wedge e_{j,k'}$ = false (Determinism check: The conjunction of the conditioned events in each row of the table is false).
4. For all $m \in M_{\mu(i)}$, there exists j such that $m_j = m$ or there exist j and k such that $m_{j,k} = m$ (Completeness check: Each mode in the mode class is in either ρ_i' s domain or its image).

These properties and assumptions on input events guarantee that ρ_i is a function. It is easy to show that a mode transition table with the format shown in Figure II.4 can be expressed in the format of an event table. Hence, a mode transition table can be expressed as an event table function F_i.

Reachability check is not used SCR table, but used ISO table newly proposed in this paper. Figure III.1 is the typical format of ISO table. ISO table is composed of 3 columns that are included Input, State and Output. Each mode of system must has input and output variables or semantic syntax. Thus, A set of input, mode(state), output is filled with ISO tables. In order to check the reachability, we apprehend the state transition using Statechart diagram first, compare input of state with output of that, finally we can find the errors that every mode(state) is not reachable.

Input	State	Output
$input_{1,1}$ $input_{1,2}$	Mode 1	$output_{1,1}$ $output_{1,2}$
$input_{2,1}$ $input_{2,2}$	Mode 2	$output_{2,1}$ $output_{2,2}$
…	…	…

Fig III.1 Typical format of ISO table

IV. Application: Water Level Monitoring System (WLMS)

IV.1 System Requirements

The requirements specification of a water level monitoring system (WLMS) used in this application comes from [8]. The system consists of two modeclasses: one that describes system behavior when the system is operating correctly and one that describes the behavior when the system is failed. But we only studied the OPERATING modeclass because FAILURE modeclass is simple and uninteresting. OPERATING modeclass is comprised of four modes.

OPERATING: The system is running.
STANBY: The system has been stopped, but has not failed.
SHUTDOWN: The water level has fallen outside allowable limits, which may cause the system to half if the water level is not restored to within the hysteresis range within 200ms.
TEST: The system is tested.

The system always starts in mode STANDBY.

IV.2 Visual Modeling of WLMS using Statechart

This section describes the Statechart visual models representing the WLMS system requirement. The Statechart diagrams are modeled by Statemate[1] that is visual modeling tool. Figure IV.1 shows the Statechart diagram of WLMS

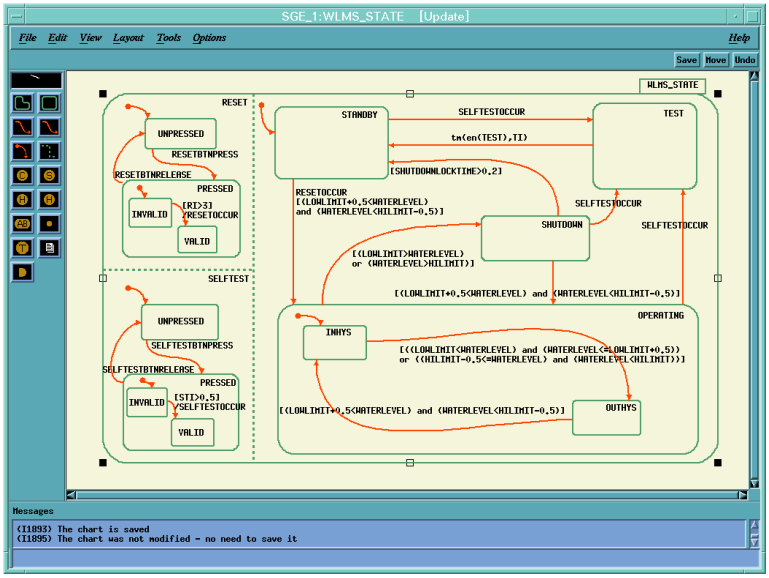

Fig IV.1 The Statechart diagram of WLMS

IV.3 SCR Specification Converted from Statechart

This section describes the SCR specification converted from Statechart diagram. SCR specifications are constructed by using the converting procedures as mentioned in section III.3. Figure IV.2 shows the mode transition table and condition table of WLMS.

IV.4 Formal Checks

In this section, we check the WLMS requirement specification using the checklists listed III.5 manually. Checking procedures of each checklist is represented in section III.5 As the result of verification, among the checklists, errors are not found in coverage, disjointness, determinism and completeness check, but in reachability check, one error is detected. In reachability checking, we use the ISO table mentioned above. In Fig IV.3, the STANDBY state has the variable of WaterLevel in Output, but not in Input. Thus, STANDBY state has the possibility that no transition is occurred when the WLMS system is STANDBY state.

Old Mode	Event	New Mode
STANDBY	@T(ResetInterval) WHEN InsideHys Range=TRUE and Within Limits=TRUE	OPERATING
	@T(SlfTst Interval) WHEN SlfTst Pressed=TRUE	TEST

Old Mode	Event	New Mode
OPERATING	@F(Within Limits) WHEN InsideHys Range=FALSE and SlfTst Pressed=FALSE and SlfTst Interval=FALSE	SHUTDOWN
	@T(SlfTst Interval) WHEN SlfTst Pressed=TRUE	TEST

Old Mode	Event	New Mode
SHUTDOWN	@T(InsideHys Range) WHEN Within Limits=TRUE and SlfTst Pressed=FALSE and SlfTst Interval=FALSE and Shutdown LockTime=FALSE	OPERATING
	@T(Shutdown LockTime) WHEN SlfTst Pressed=FALSE and SlfTst Interval=FALSE	STANDBY
	@T(SlfTst Interval) WHEN SlfTst Pressed=TRUE	TEST

Old Mode	Event	New Mode
TEST	@T(TestInterval)	STANDBY

Mode	Condition	
Shutdown	WaterLevel<LowLevel and WaterLevel>HighLevel	LowLevel ≤ WaterLevel ≤ HighLevel
Alarm	On	Off

Fig IV.2 SCR specification of WLMS

Input	State	Output
TestInterval ShutdownLockTime	STANDBY	ResetInterval ResetBtnPressed WaterLevel
ResetInterval ResetBtnPressed WaterLevel	OPERATING	WaterLevel SlfTestBtnPressed SlfTestInterval
WaterLevel	SHUTDOWN	ShutdownLockTime WaterLevel SlfTestBtnPressed SlfTestInterval
SlfTestBtnPressed SlfTestInterval	TEST	TestInterval

Fig IV.3 ISO table of WLMS

V. Conclusion and Future Works

In this work, a formal requirement analysis method for Nuclear Power Plant (NPP) Instrumentation and Control (I&C) systems is proposed. This method use the Statechart diagram, Software Cost Reduction (SCR) formalism and ISO table newly suggested in this paper for checking the modeled systems formally. The combined method of utilizing Statechart, SCR and ISO table has the advantage of checking the system easily, visually and formally.

This method is applied to the Water Level Monitoring System (WLMS). As a result of the formal check, one reachability error is detected.

In the future work, we make a plan to apply the last checklist that is not applied in this paper, which is backward hazard analysis. Additionally, we are planning to use this formal requirement analysis method in real NPP components, such as protection system and monitoring system since this method can be highly time and cost effective for detecting errors in requirement specification safety analysis.

References

[1] “Statemate 4.5 User Reference Manual”, I-Logix Inc. Burlington, MA, Aug. 1992.

[2] Kyo C. Kang and Kwang I. Ko “Formalization and Verification of Safety Properties of Statechart Specifications”, IEEE Software Engineering Conference 1996 Proceedings, Asia Pacific, Dec. 1996 pp. 16-27.

[3] Constance Heitmeyer and Bruce Labaw, “Consistency Checking of SCR-Style Requirements Specification”, International Symposium on Requirements Engineering, March, 1995.

[4] D.Harel, “Statecharts: A visual formalism for complex systems”, Sci. of Comput. Prog., Vol. 8, pp. 231-274, 1987.

[5] D. Craigen et al. “An international survey of industrial applications of formal methods.”, Technical Report NRL-9581, NRL, Wash., DC, 1993.

[6] Jaffe, M. S., Leveson, N. G. Heimdahl, M. P. E., and Melhart, B. E., Software Requirements Analysis for Real-Time Process-Control Systems, IEEE Trans. Software Engineering, 17, 241-258 (1991).

[7] Robyn R. Lutz, “Targeting Safety-Related Errors During Software Requirements Analysis.” J. Systems Software 1996; 34; 223-230.

[8] J. van Schouwen, “The A-7 requirements model: Reexamination for real-time systems and an application to monitoring systems,” Dept. of Computing and Information Science, Queens’ s University, Kingston, Ontario, Canada, Tech. Rep. TR-90-276, May 1990.