# 반응형 실시간 소프트웨어를 명세하고 분석하기 위한 기법

오윤주, 조재명, 유준범, 차성덕
한국과학기술원 전기전자 및 전산학과 전산학 전공
{yjoh, jmcho, jbyoo, sdcha}@salmosa.kaist.ac.kr

# A Technique to Specify and Analyze Reactive and Real-Time Software

Younju Oh, Jaemyoung Cho, Junbeom Yoo, Sungdeok Cha
Div. of CS, Dept. of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology

**Abstract**

Writing requirements in formal notation for a safety-critical system can improve software quality and reduce the errors that may arise later on in the software development life cycle. In this paper, we propose a formal specification approach used to describe the nuclear control system. The approach is based on the existing AECL approach that was the only formal specification technique applied to nuclear control systems in the past. Although the approach is AECL-based, the complex descriptions of certain requirements have been reduced by using different specification techniques. We discuss the differences and how the proposed approach provides not only specification but also verification environment.

## 1. Introduction

There is a growing need for reliable methods in designing correct and safe systems. In safety-critical systems, such as nuclear control systems, inconsistencies within the description of the system causes accidents to occur which results in costly damages. For such systems to be safe and reliable, inconsistency in describing and specifying requirements should be avoided by using formal specification approaches in the software requirements specification (SRS).

Many specification languages are suggested to specify reactive and real-time systems, but it is difficult to find a suitable language for nuclear control systems. A nuclear system is an extremely safety-critical system, and stable technologies should be applied to it.

In this paper, we propose a technique to specify and analyze reactive and real-time software that provides environment to verify the functional requirements of a nuclear control system. Specifying requirements in a formal notation allows such properties as unambiguity, consistency, and completeness of the SRS. However, verifying the properties derived from the requirements still remains difficult. The existing techniques[1] are time consuming and require manual effort to verify the properties of the system. The proposed technique provides not only specification approach in specifying requirements but also verification environment.

The proposed specification approach is based on the existing AECL approach[2]. It shares the same notation of describing system requirements in Function Overview Diagram (FOD) and in describing each function in tabular notation using the Structured Decision Table (SDT). However, the main purpose of our approach is to reduce the specifying complexity of the AECL approach. The AECL approach describes all requirements specifications based on function nodes in FOD and tables in SDT, which makes timing requirements and history related requirements difficult to specify, whereas the proposed approach uses automata and timed-automata to specify such behaviors that are not easily expressed with the notations of FOD and SDT.

This paper is organized as follows. Section 2 gives an overview of the existing AECL approach. Section 3 introduces the proposed approach and a detailed description of how it differs from the existing AECL approach. A brief introduction on the editor made for this approach is also included in this section. In section 4, we present some conclusions and future work.

## 2. AECL Approach

The Atomic Energy of Canada Limited (AECL) approach specifies a methodology and format for the specification of software requirements for safety critical software used in real-time control and monitoring systems in nuclear generating systems. It is a SCR-style SRS verification method based on Parnas' four variable method. A system reads environment states through monitored variables that are transformed into input variables. The output values of the output variables are calculated and are changed into control variables.

The AECL provides two different views of the requirements. A larger view is the FOD and each of the function in it is described by the smaller view of the SDT.

The AECL approach specifies all requirements of the nuclear control system in the FOD and SDT notations. This is somewhat complex in cases where timing requirements and history related requirements are considered. This difficulty of specification is modified in our approach, which is discussed later on in this paper.

An FOD is a similar notation to DFD (Data Flow Diagram). However, it not only shows the data flows but also hierarchies among the functions and the groups of functions. The data values are also computed by the dependencies between the data flow. The following Fig. 1 is part of an FOD description of the SRS in the Wolsong SDS2 (Shutdown System2)[3]. There are three nodes that represent functions, arrows that show the flow of

input/output data and *s_PDLCond* that is a state variable. Other notations include inputs and outputs from the computer system described in rectangles and timing functions described in vertical bars.

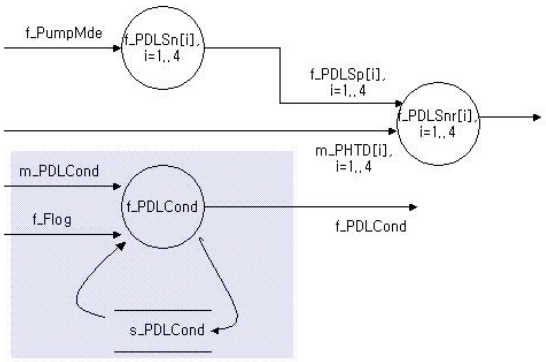


Fig. 1 FOD

The notation to specify the lowest level, a function, is the SDT which is an AND_OR table. The required behavior of each function is described in a tabular notation as shown below. Fig. 2 is the SDT for the function *f_PDLCond* specified in Fig. 1 with the details of the SDT included.

**Description :**
Determine the PHT low core differential pressure immediate trip conditioning status

**Function Inputs :**
m_PDLCond
f_Flog
s_PDLCond

**Condition Macros :**

w_FlogPDLCondLo[f_Flog]

| | |
|---|---|
| a | f_Flog<k_FlogPDLLo-k_CondHys |
| b | k_FlogPDLLo-k_CondHys<=f_Flog<k_FlogPDLLo |
| c | f_Flog>=k_FlogPDLLo |

w_FlogPDLCondHi[f_Flog]

| | |
|---|---|
| a | f_Flog<k_FlogPDLHi-k_CondHys |
| b | k_FlogPDLHi-k_CondHys<=f_Flog<k_FlogPDLHi |
| c | f_Flog>=k_FlogPDLHi |

| **CONDITION STATEMENTS** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| m_PDLCond=k_CondSwLo | T | T | T | T | F | F | F | F |
| w_FlogPDLCondLo[f_Flog] | a | b | b | c | - | - | - | - |
| w_FlogPDLCondHi[f_Flog] | - | - | - | - | a | b | b | c |
| s_PDLCond=k_CondOut | - | T | F | - | - | T | F | - |
| **ACTION STATEMENTS** | | | | | | | | |
| f_PDLCond=k_CondOut | X | X | | | X | X | | |
| f_PDLCond=k_CondIn | | | X | X | | | X | X |

Fig. 2 SDT of f_PDLCond

In Fig. 2, the function *f_PDLCond* produces either output *k_CondOut* or *k_CondIn* depending on the condition statements in the SDT. The condition statements are AND related. For example, in the first column of the SDT, if the condition *m_PDLCond=k_CondSwLo* is true and if the condition *w_FlogPDLCondLo[f_Flog]* satisfies the condition macro 'a', then the function *f_PDLCond* produces the output *k_CondOut* which can be seen in the related action statements.

## 3. The Proposed Approach

The proposed approach is an extended formal verification method of the existing SCR-style AECL approach. The specification approach was originally designed to simplify the complex specification techniques of certain requirements in the AECL approach. It is an improved method in describing behavior of the history related requirements and timing requirements of the nuclear control system by specifying them in automata and timed-automata respectively. In the existing AECL method, all specifications including history related requirements and timing requirements are specified with only one type of function node in the FOD and with SDT tables. However, our approach uses three different types of nodes in the FOD to specify the properties derived from the requirements. The types consist of nodes that specify history related requirements that are described in automata[4], timing requirements that are described in timed-automata[5], and nodes that specify all other requirements exclusive of the previous two types of functional requirements.

### 3.1 Specifying History Related Requirements

The history related requirement of the system is the specification of the previous state or value that a function or functions must have before the next transition can occur. For example, in the FOD of the AECL approach in Fig. 1, the previous state of the function *f_PDLCond* is *s_PDLCond*, which is shown with two horizontal bars beneath the *f_PDLCond* node. This requirement is shown in Fig.2 on the fourth row of the condition statements. However, our approach introduces a much simpler way by using automata to describe the history related behaviors that are difficult to specify with functions as did in the AECL approach. The following Fig. 3 is the automata of the function *f_PDLCond* in Fig.1 and Fig 2 described in the proposed approach.

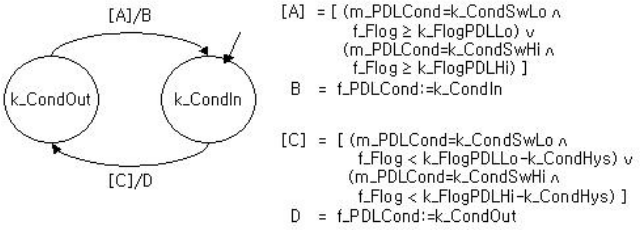


Fig. 3 Automata Description for f_PDLCond

The conditions required to go to *k_CondOut* state or *k_CondIn* state and the action that occurs when the conditions are satisfied, are shown with the corresponding transition.

### 3.2 Specifying Timing Requirements

Functional timing specifications represent timing requirements that are an integral part of the actual function. An example of a time related functional requirement is that "function must maintain a light 'on' for ten seconds before the activator goes off". The timing interval would then be ten seconds. The proposed approach describes the timer function in timed automata instead of the complex timer function used in the AECL approach shown in Fig. 4. Vertical bars are timing functions, *s_pending* is the state function and *t_Pending* is a timing function for describing the time delay. *t_Trip* is also a timing function for describing the required trip duration and *f_PDLDly* is a

function with history requirements.

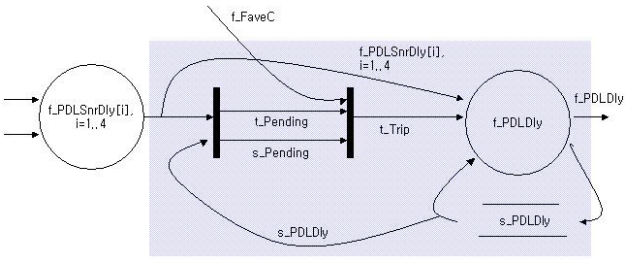


Fig. 4 Timing specifications in FOD AECL approach

Timing and history related requirements as this must be clearly stated in the SRS. Fig. 4 uses only one type of node to define all requirements. However, in the FOD below, the timing and history related requirements which is specified in the box of Fig. 4 is specified into one type of node, the *th_PDLDly* function node. The "*th_*" is for the timed-history node.

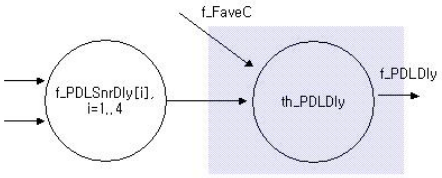


Fig. 5 FOD in the proposed approach

The following Fig. 6 is the completed timed-automata obtained from the requirements specification of the FOD in Fig. 5 and from the related SDT specification.

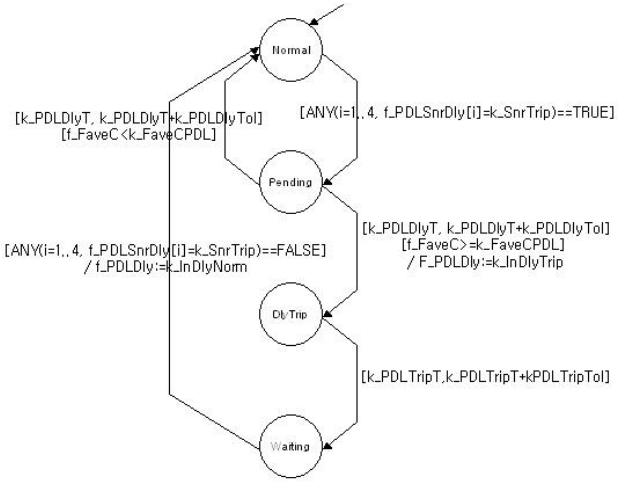


Fig. 6 Timed-Automata

### 3.3 Verification

Verifying the properties of the nuclear system is very time consuming and takes much effort using the existing AECL approach. However, our approach allows the SRS of the nuclear control system to be verified in an automated environment. The SRS in the proposed approach is converted into XML, which a converting tool automatically transforms into PVS specification language. The translated specification is then verified with the PVS theorem prover[6].

### 3.4 The Editor

The Editor for our approach is a platform independent tool made with JAVA for formally specifying the SRS of the nuclear control system. It provides environment to draw FOD and SDT and allows automata diagrams to be built from the nodes of the FOD. The Editor also gives a hierarchical view of the SRS described as can be seen on the left side of Fig.7.

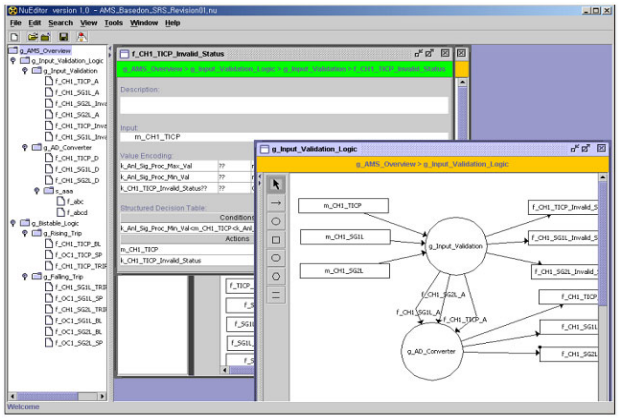

Fig. 7 Editor

## 4. Conclusions and Future Work

We have presented a formal specification method which is simple yet more suitable for specifying software requirements of nuclear control systems. First, we have described that our approach is based on the existing AECL approach with better techniques for specifying history and timed related properties that are the main motivation for designing the proposed approach. Second, we have shown that verifying the properties derived from the system requirements can be done more easily using the PVS theorem proving in the given environment.

There are possible directions that the proposed approach can be developed toward for future work. It will be extended so that verification of the specification can be done with model checking using SMV or SPIN. Our approach will be developed further on so that it will be a more complete approach in specifying the requirements of the safety-critical nuclear control system.

## References

[1] WolsongnNPP 2/3/4, "Software Verification Report Code Verification, SDS2 PDC," *Design Document no. 86-68350-SVR-002, Rev. 0*, June 1994.

[2] WolsongnNPP 2/3/4, "Software Work Practice Procedure for the Specification of SR for Safety Critical Systems," *Design Document no. 00-68000-SWP-002, Rev. 0*, Sept. 1991.

[3] WolsongnNPP 2/3/4, "Software Requirements Specification for Shutdown Systems 2 PDC," *Design Document no. 86-68350-SRS-001, Rev. 0*, June 1993.

[4] J. Hopcroft and J. Ullman, "Introduction to Automata Theory," *Language and Computation*, Addison-Wesley, 1979.

[5] R. Alur and David L. Dill, "A theory of Timed Automata," *Theoretical Computer Science* Vol. 126, No. 2, pp. 183-236, April 1994.

[6] T. Kim, D. Stringer-Calvert, and S. Cha, "Formal Verification of Functional Properties of an SCR-Style Software Requirements Specification Using PVS," *TACAS 2002, LNCS 2280*, pp. 205-220, 2002.