A Formal Software Requirements Specification Method for Digital Plants Protection Systems

Junbeom Yoo       Sungdeok Cha
KAIST             KAIST

CS/TR-2003-191
June 1, 2003

KAIST
Department of Computer Science
A Formal Software Requirements Specification Method for Digital Plants Protection Systems

Junbeom Yoo  
KAIST

Sungdeok Cha  
KAIST

Abstract

This article describes NuSCR, a formal software requirements specification method for digital plant protection system in nuclear power plants. NuSCR improves the readability and specifiability by supplying different notations on the basis of the typical operation categories. The characteristics of the software process controller in nuclear power plants, s.t. periodic sequential processing and classifiable operations, makes this possible. We introduce the syntax and semantics of NuSCR in the paper. An ATWS mitigation system in Korean nuclear power plant is used as a case study to illustrate the usefulness of NuSCR.

1 Introduction

Software safety is an important property for safety critical systems, especially those in aerospace, satellite and nuclear power plants, whose failure could result in danger to human life, property or environment. It is recently becoming more important due to the increase in the complexity and size of safety critical systems [9]. Formal software requirements specification is known as a means to increase the safety of such safety critical systems in the early phase of software development process. It guides the developer to specify all requirements explicitly without any assumptions or omissions. Also many recognized formal verification methods, such as model checking [2] [3] and mechanized theorem proving [21], can be applied to the formal software requirements specification.

In the area of nuclear power plant systems, the formal specification of software requirements becomes more important with the replacement of existing analog systems by digital systems composed of software process controllers [12]. Nowadays, software requirements and design specifications which are suitable for the characteristics of nuclear power plants system, are becoming new research issues by many researcher as the Practical Formal Specification(PFS) project in aerospace applications[11].

Typical characteristics of digital protection controllers in nuclear power plant systems are as follows. First, numerous inputs are calculated by the software process controllers. To maintain the system to be safe, all the status of reactors and peripherals, i.e. turbines, steam generators, and other subsystems, should keep being observed. Second, the software operates sequentially, s.t. receives software inputs, calculates with them, and then emits software outputs. It repeats the sequential operation periodically at every predefined time interval. Last of all, all the possible operations of the software process controller can be classified into three categories. They are function-based, state-based, and timing-based operations. Function-based operations are the functions that gets inputs, calculates with inputs only, and then emits an output. State-based operations are the operations that require the history information additionally. Timing-based operations are the ones which require timing constraints in addition to the history information.

NuSCR is a formal software requirements specification extended from SCR(Software Cost Reduction) [5] to easily specify the functional requirements of safety critical software, especially those
of nuclear power plants. It is based on Parnas’ Four-Variable Model [13] and uses FOD(Function Overview Diagram) for the overview of data flows in the same way as [22]. [22] is a variant of SCR, which was proposed by AECL(Atomic Energy of Canada Limited) and was used for the formal software requirements specification for SDS2(ShutDown System 2) in Wolsong nuclear power plant in Korea. NuSCR improves the readability and specifying ability by supplying different notations on the basis of their typical operation categories. The characteristics of software process controllers in nuclear power plants, s.t. periodic sequential processing and classifiable operations, makes this possible.

In the approach of AECL, the state-based operations such as trip set point hysterisis are specified by functions, although they have originally state-based features. It is because that the basic specifying concept of AECL approach is to specify all aspects by functions. Timing-based operations such as delay timer are also specified by special timer functions, which are too hard to define and understand. In NuSCR, we adopts FSM(Finite State Machine) for specifying state-based parts, and a kind of TTS(Timed Transition System)[18] for timing-related parts in software requirements.

NuSCR formal software requirements specifications can be verified by theorem prover PVS [15] with our approach [7] developed for SCR. Using PVS, we can verify the structural properties such as input/output completeness, consistency, and circular dependencies in NuSCR specification. NuSCR specifications can also be verified by model checker such as the SMV [10], based on the formal semantics of NuSCR presented in this paper. We are developing an automatic translator that translates NuSCR specification into SMV inputs.

The remainder of the paper is organized as follows: Section 2 reviews SCR and the variant proposed by AECL. Section 3 introduces the specification constructs in NuSCR. In Section 4, we represent the formal semantics of NuSCR software requirement specifications. We then briefly introduce NuSCR requirements specification for AMS(ATWS Mitigation System) as a case study, and describe the software development environment in progress in Section 5. Conclusion and future work direction are in Section 6.

2 Formal Requirements Specification Approaches

Some formal requirements specification methods such as Z [16], VDM [6], and Larch [4] focus on specifying the behavior of sequential systems. These approaches use rich mathematical structures like sets, relations, and functions to describe states and use pre-conditions and post-conditions for state transitions. However, these approaches are too expressive to specify nuclear power plants software concisely.

SCR [5] was introduced more than twenty years ago to specify the software requirements of real-time embedded systems. Recently it has been extended to incorporate both functional and non-functional(e.g. timing and accuracy) requirements [14], [1]. As it was designed to be used by engineers, the SCR methods has been successfully applied to a variety of practical systems, such as the A-7 Operational Flight Program [17], submarine communication system, and safety-critical component of Darlington nuclear power plant in Canada [1].

The approach [1] applied to the Darlington nuclear power plant by AECL is the first attempt as the formal software requirements specification for nuclear power plants system and it was also applied to SDS2(ShutDown System 2) in Wolsong nuclear power plant in Korea [23]. The approach is based on SCR and has some extensions from SCR. At first, to specify the software more concisely, it combined the three tables of SCR, the mode transition table, event table, and condition table, into a table called SDT(Structured Decision Table). It uses FOD(Function Overview Diagram) which is similar to DFD(Data Flow Diagram) for the overview of the system. Finally, it provides sophisticated functions for describing precision and tolerance to describe timing con-
The characteristics of AECL approach is as follows: (i) SDTs and FODs are familiar notations for domain engineers and developers. (ii) However, SDTs are too complicated. There usually are too many columns and rows to understand. The complexity of SDTs comes from the basic idea of the AECL approach. Although they originally have state-based features, the function form of them makes unnecessary complication arise. (iii) Managements of time-related features such as timers are too complicated to define and understand. They use the special timing functions for specifying time-related requirements. However the definition of them is too hard to be known by common domain engineers by intuition.

3 NuSCR Software Requirements Specification Constructs

NuSCR basically uses four constructs, monitored variable, input variable, output variable, and controlled variable according to Parnas’ Four-Variable Model[10]. In addition, to specify the relations of Parnas’ Four-Variable Model in practical and domain dependent manners, we introduce three other basic constructs, function variable, history variable, and timed history variable. These three constructs can be defined as SDT, FSM, and TTS respectively. The relationship of all constructs is represented by FOD.

Naming Convention NuSCR uses the prefix naming convention as follows to distinguish each construct efficiently. Two prefixes, "g" and "k", are introduced for the convenience of specification:

- m: monitored variable
- i: input variable
- f: function variable
- h: history variable
- th: timed history variable
- g: set of function variable, history variable, or timed history variable
- k: predefined constant
- o: output variable
- c: controlled variable

System Entities System entities constructing NuSCR software requirements specification are defined as follows:

- V_I: a set of system input variables
- V_F: a set of function variables
- V_H: a set of history variables
- V_TH: a set of timed history variables
- V_O: a set of system output variables
- V_SE:
- a set of system entities
- \( V_I \cup V_F \cup V_H \cup V_{TH} \cup V_O \)

- \( D_{SE} \): a set of all possible domain for every \( r \) in \( V_{SE} \)
- \( \sigma \): a valuation function that maps \( V_{SE} \) into \( D_{SE} \)

- \( \sigma[d/v] \) means that (let \( v \in V_{SE}, V_v = V_{SE} - \{v\}, d \in D_{SE} \))
  - \( \sigma[v] = d \) and \( \sigma[V_v] = \sigma[V_v] \)
  - \( \sigma(f(v)) = f(\sigma[v]) = f(v)(\sigma) \)

**Condition Statements**  Condition statements are the predicates on the value of all entities in \( SE \). The condition statements in NuSCR are defined as BNF form as follows:

Let \( r \in V_{SE}, v_r \in D_{SE}, a, b \in N, \text{ and } \odot \in \{\neq, \leq, <, \geq, >\} \),

- simple_condition := \( r \odot v_r \) | \( r \odot r \) | TRUE | FALSE
- complex_condition := simple_condition \& simple_condition
- timed_condition := [\( a, b \)] complex_condition

As the above definition, \( timed\_condition \) is a \( complex\_condition \) appended by the timing constraints \( [a, b] \) which means a duration of time \( a \) and \( b \). \( timed\_condition \) is used in defining timed history variables, and \( complex\_condition \) is used in defining both function variables and history variables.

**Assignment Statements**  Assignment statements mean the valuation of entities in \( SE \). The assignment statements in NuSCR are defined as BNF form as follows:

Let \( r \in V_{SE}, v_r \in D_{SE}, a, b \in N, \text{ and } \oplus \in \{+, -, *, \div\} \),

- assignment := \( (r := v_r) \) | \( (r := r) \) | \( (r := r \oplus r) \) | \( (r := r \oplus v_r) \)

**Function Variable**  Function variables are used for specifying the mathematical functional behavior of a system. They are defined as SDTs. SDT is a kind of Condition/Action table, which represents the actions(assignment statements) performed if their guiding conditions(condition statements) are satisfied. Tabular notations such as SDTs have the merit of being familiar to engineers and developers. Conditions in SDT are the \( complex\_conditions \) with the inputs of the function variable. Actions are the \( assignment \) to the function variable itself.

<table>
<thead>
<tr>
<th>Conditions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( th_{CH1_TICP_TRIP_Status} = k_{CH1_TICP_TRIP} )</td>
<td>T F</td>
</tr>
<tr>
<td>( th_{CH1_TICP_TRIP_Status} = k_{CH1_TICP_NORMAL} )</td>
<td>F T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{OC1_TICP_BL_I} := f_{OC1_TICP_I} \times 100 )</td>
<td>X</td>
</tr>
<tr>
<td>( f_{OC1_TICP_BL_I} := f_{OC1_TICP_I} \times 10 )</td>
<td>X</td>
</tr>
</tbody>
</table>

Figure 1: Structured decision table for \( f_{OC1\_TICP\_BL\_I} \)
(Fig. 1) is an SDT defining function variable \( f_{\text{OCl \_TICP \_BL \_I}} \). It is excerpted from NuSCR software requirements specification for AMS[20]. Input entities for this variable are \( \text{th \_CH1 \_TICP \_TRIP \_STATUS} \) and \( f_{\text{OCl \_TICP \_I}} \). The detailed interpretation of such definition is as follows:

\[
\text{If th\_CH1\_TICP\_TRIP\_STATUS is same as } k_{\text{CH1\_TICP\_TRIP}} \text{ and not same as } k_{\text{CH1\_TICP\_NORMAL}}, \text{ then the new value for } f_{\text{OCl\_TICP\_BP\_I}} \text{ is } f_{\text{OCl\_TICP\_I}} \text{ multiplied by } 100. \text{ Else if th\_CH1\_TICP\_TRIP\_STATUS is same as } k_{\text{CH1\_TICP\_NORMAL}} \text{ and not same as } k_{\text{CH1\_TICP\_TRIP}}, \text{ then the new value for } f_{\text{OCl\_TICP\_BP\_I}} \text{ is } f_{\text{OCl\_TICP\_I}} \text{ multiplied by } 10.
\]

**History Variable** History variables are used for specifying the state-based behavior of a system. They are defined as FSMs. FSM consists of finite number of states, transitions between states, and labels on each transition. Labels are the Conditions/Actions statements which are same as that of SDTs. Conditions in FSM’s transition labels are the complex conditions with the inputs of the history variable. Actions are the assignment to the history variable itself. If the transition condition is satisfied in the current state, then the action is performed and the state transition occurs.

(Fig. 2) is a FSM defining history variable \( h_{\text{OCl\_TICP\_SP\_I}} \). Input entity for this variable is \( \text{th\_CH1\_TICP\_TRIP\_STATUS} \) and the initial state is NORMAL. The detailed interpretation of such definition is as follows:

In state NORMAL, if \( \text{th\_CH1\_TICP\_TRIP\_STATUS} \) is same as \( k_{\text{CH1\_TICP\_TRIP}} \), then the new value for \( h_{\text{OCl\_TICP\_SP\_I}} \) is \( h_{\text{OCl\_TICP\_SP\_I}} \) minus \( k_{\text{TICP\_HYS\_SP}} \) and transition to state TRIP occurs. Also in state TRIP, if \( \text{th\_CH1\_TICP\_TRIP\_STATUS} \) is same as \( k_{\text{CH1\_TICP\_NORMAL}} \), then the new value for \( h_{\text{OCl\_TICP\_SP\_I}} \) is \( h_{\text{OCl\_TICP\_SP\_I}} \) plus \( k_{\text{TICP\_HYS\_SP}} \) and transition to state NORMAL occurs.

**Timed History Variable** Timed history variables are used for specifying the time-related behavior of system. They are defined as a kind of TTS [18]. TTS is a FSM extended with the timing constrains \([a, b]\) in transition conditions. \([a, b]\) means the time duration between time \(a\) and \(b\).

(Fig. 3) is a TTS defining timed history variable \( \text{th\_CH1\_TICP\_TRIP\_STATUS} \). Input entities for this variable are \( f_{\text{OCl\_TICP\_I}} \) and \( h_{\text{OCl\_TICP\_SP\_S}} \) and the initial state is NORMAL. The detailed interpretation of such definition is as follows:

\[
\begin{align*}
\text{th\_CH1\_TICP\_TRIP\_STATUS} &= k_{\text{CH1\_TICP\_NORMAL}} \\
\langle h_{\text{OCl\_TICP\_SP\_I}} := f_{\text{OCl\_TICP\_I}} - k_{\text{TICP\_HYS\_SP}} \rangle &\quad \text{if th\_CH1\_TICP\_TRIP\_STATUS is same as } k_{\text{CH1\_TICP\_TRIP}}, \text{ transition to state TRIP occurs.}
\end{align*}
\]
In state NORMAL, if $\text{l}\_\text{OC1\_TICP\_I}$ is greater than $\text{h}\_\text{OC1\_TICP\_SP\_S}$, then immediate transition to state WAITING occurs and the output value is same as the previous one. If the condition is satisfied for 10 seconds from the point entering state WAITING, then the new value for $\text{th}\_\text{CH1\_TICP\_TRIP\_STATUS}$ is $\text{k}\_\text{CH1\_TICP\_TRIP}$ and transition to state TRIP occurs. Else if the condition is not satisfied for 10 seconds from the point entering state WAITING, then the new value for $\text{th}\_\text{CH1\_TICP\_TRIP\_STATUS}$ is $\text{k}\_\text{CH1\_TICP\_NORMAL}$ and transition to state NORMAL occurs. In state TRIP, if $\text{l}\_\text{OC1\_TICP\_I}$ is less than $\text{h}\_\text{OC1\_TICP\_SP\_S}$, then immediate transition to state NORMAL occurs and the new value for $\text{th}\_\text{CH1\_TICP\_TRIP\_STATUS}$ is $\text{k}\_\text{CH1\_TICP\_NORMAL}$.

**Function Overview Diagram**

FOD(Function Overview Diagram) is a kind of DFD, which describes the relationship between constructs in $V_{SE}$ in NuSCR software requirements specification. Each construct in $V_{SE}$ is represented by specific nodes, and the relationship between them is represented by unidirectional arrows. FOD is composed hierarchically and in this case the group nodes are used. The name of each node is followed by the prefix naming convention described above.

(Fig. 4) is a FOD for group node $g\_\text{Bistable\_Logic}$. It is composed of three group nodes $g\_\text{Rising\_Trip}$, $g\_\text{Falling\_Trip}$, and $g\_\text{Digital\_Trip}$. As shown in (a), the group node $g\_\text{Rising\_Trip}$ has $\text{l}\_\text{OC1\_TICP\_I}$ and $\text{l}\_\text{OC1\_TICP\_BP\_S}$ as input entities. The output of $g\_\text{Rising\_Trip}$ are $\text{l}\_\text{OC1\_TICP\_BL\_I}$, $\text{th}\_\text{CH1\_TICP\_TRIP\_STATUS}$, and $\text{h}\_\text{OC1\_TICP\_SP\_I}$. The refined FOD is represented in (b). It is composed of three nodes which have corresponding outputs respectively. The relationship between these three nodes are represented by arrows.

4 Semantics for NuSCR

The behavior of software system specified by NuSCR can be defined based on the behavior of FOD.

**Function Overview Diagram**

FOD in NuSCR is defined as a tuple:

$$FOD = \langle N, T, I, O \rangle$$
Figure 4: Function overview diagram for $g_{\text{Rising Trip}}$
• $N$
  - a set of all function nodes in FOD
  - all nodes in $V_F$ and $V_H$ and $V_{TH}$ are defined as functions
• $T$
  - a set of transition $(n_1, n_2)$ between all nodes $n_1, n_2$ in $N$
  - $\forall t = (n_1, n_2) \in T$, $n_1$ has a precedence on $n_2$
• $I$
  - a set of input values into FOD
  - It is mapped into input variables $V_I$ of FOD
• $O$
  - a set of output values from FOD
  - It is mapped into output variables $V_O$ of FOD

From the definition of FOD above, FOD can be defined as a function $f_{FOD}$ from input values $I$ to output values $O$. Also as all nodes in $N$ have partial ordering according to the transition in $T$ and all nodes are defined as functions, $f_{FOD}$ can be represented as a function composition of all nodes in $N$ according to the partial orders on their precedence.

$$f_{FOD}(\sigma[I/V_I]) = \sigma[O/V_O]$$
$$f_{FOD} = \text{(mathematical function composition by partial orders on their precedence)}$$

**Function Variable Node** Function variable in NuSCR is represented by a function variable node in FOD. It is defined by SDT. Let $I_{FV}$ be the set of input values from other nodes in FOD into the function variable node itself. Let $O_{FV}$ be the set of output values from this node. They can be mapped into the set of variables, $V_{FI}$ and $V_{FO}$ respectively. Then complex conditions in SDT are the predicate on $V_{FI}$, and actions are the assignments on $V_{FO}$ which is the function variable itself.

SDT is defined as a set of a pair $(p, a)$, where $p \in P$ and $a \in A$. $P$ is a set of boolean predicates on $V_{FI}$, which is the conjunction of complex conditions in condition statements and corresponding boolean conditions. $A$ is a set of assignments to $V_{FO}$ which is just the function variable itself.

$$SDT : \text{a set of pair } (p, a)$$
  1. $p \in P$ and $a \in A$
  2. $\forall (p, a) \in SDT$, $p(\sigma) = T$ then $a(\sigma) = \sigma[O_{FV}/V_{FO}] = \sigma'$

For example, SDT in (Fig. 5) can be defined as follows:

$$SDT = \{(\text{Cond}_1 = T \land \text{Cond}_2 = F, \text{Assign}_1), (\text{Cond}_2 = T \land \text{Cond}_3 = F, \text{Assign}_2), (\text{Cond}_1 = T \land \text{Cond}_3 = T, \text{Assign}_3)\}$$

From the definition of SDT above, a function variable node can be defined as a function $f_{FV}$ with input values $I_{FV}$ to output values $O_{FV}$ as follows.
<table>
<thead>
<tr>
<th>Conditions</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cond 1</td>
<td>T</td>
<td>-</td>
<td>T</td>
</tr>
<tr>
<td>Cond 2</td>
<td>F</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td>Cond 3</td>
<td>-</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign 1</td>
<td>X</td>
</tr>
<tr>
<td>Assign 2</td>
<td>X</td>
</tr>
<tr>
<td>Assign 3</td>
<td>X</td>
</tr>
</tbody>
</table>

Figure 5: Structured decision table for a function variable

\[
f_{FV}(I_{FV}) = \sigma[I_{FV}/V_{FI}] = O_{FV}\]

**History Variable Node**

History variable in NuSCR is represented by a history variable node in FOD. It is defined by FSM which is composed of states, transitions between states, and labels on transitions. Let \(I_{HV}\) be the set of input values from other nodes in FOD into the history variable node. Let \(O_{HV}\) be the set of output values from this node. They can be mapped into the set of variables, \(V_{HI}\) and \(V_{HO}\) respectively. Then complex conditions in FSM are the predicate on \(V_{HI}\) and actions are the assignments on \(V_{HV}\) which is the history variable itself. FSM can be defined as a relation described below:

\[
FSM = \langle S_H, s_0, C, A, R \rangle
\]

- \(S_H\) : a set of states in history variable node
- \(s_0\) : initial state in \(S_H\)
- \(C\) : a set of complex conditions
- \(A\) : a set of assignments
- \(R\) :
  - \(S_H \times C \times A \times S_H\) is a transition relation
  - \(\forall r (s, c, a, s') \in R, \text{ s.t. } \text{current\_state} = s \text{ and } c(\sigma) = T \text{ then } a(\sigma) = \sigma[O_{HV}/V_{HO}] = \sigma'\)

\(\text{current\_state}\) in the definition above means the variable in \(CS_H\), which indicates the current state of the history node. It will be used in the definition of the overall NuSCR system. History variable in (Fig. 6) can be defined as a relation as follows:

\[
FSM = \langle S_H, s_0, C, A, R \rangle
\]

- \(S_H = \{S_1, S_2, S_3\}\)
- \(s_0 = S_1\)
- \(C = \{\text{Cond}_1, \text{Cond}_2, \text{Cond}_3, \text{Cond}_4\}\)
- \(A = \{\text{Assign}_1, \text{Assign}_2, \text{Assign}_3, \text{Assign}_4\}\)
- \(R = \{(S_1, \text{Cond}_1, \text{Assign}_1, S_2), (S_2, \text{Cond}_3, \text{Assign}_3, S_1), (S_2, \text{Cond}_2, \text{Assign}_2, S_3), (S_3, \text{Cond}_4, \text{Assign}_4, S_1)\}\)

From the definition of FSM above, a history variable node can be defined as a function \(f_{HV}\) from input values \(I_{HV}\) to output values \(O_{HV}\) as follows.
Timed History Variable Node Timed history variable in NuSCR is represented by a timed history variable node in FOD. It is defined by TTS which is a FSM extended with timing constraints \([a, b]\) in transition labels. \(a\) and \(b\) means the minimum and maximum delay in the transition respectively. Let \(I_{THV}\) be the set of input values from other nodes in FOD into the timed history variable node. Let \(O_{THV}\) be the set of output values from this node. They can be mapped into the set of variables, \(V_{THI}\) and \(V_{THO}\) respectively. Then timed conditions are the predicate on \(V_{THI}\) and timing constrains \([a, b]\), and actions are the assignment on \(V_{THV}\) which is the history variable itself. TTS can be defined as a relation described below:

\[
TTS = (S_{TH}, s_0, C, A, R)
\]

- \(S_{TH}\) : a set of states in timed history variable node \(\times lc\), where \(lc\) is a local clock in \(LC\)
- \(s_0\) : initial state in \(S_{TH}\)
- \(C\) : a set of timed conditions
- \(A\) : a set of assignments
- \(R\):
  - \(S_{TH}\times C\times A\times S_{TH}\) is a transition relation
  - \(\exists r (s, c, a, s')\) in \(R\),
    s.t. \(current\_state = s\) and \(c(\sigma) = T\) then \(a(\sigma) = \sigma[O_{THV}/V_{THO}] = \sigma'\)

The behavior of transition relations in TTS is a little different from that of FSM because of the timing constraints. For example, the transition from state \(S_1\) to \(S_2\) in (Fig. 7) has the transition labeled with "\([a, b]Cond_1/Assign_1\)". The minimum delay \(\sigma\) means that when the control of timed history node has resided at the location \(S_1\) for at least \(a\) time units during which the guard \(Cond_1\) has been continuously true, then the transition from \(S_1\) to \(S_2\) may occur. The maximum delay \(b\) means that whenever the state of history variable has resided at \(S_1\) for \(b\) time units during which the guard \(Cond_1\) has been continuously true, then the transition from \(S_1\) to \(S_2\) has to occur. The behavior of this transition can be described as follows. "\(lc := lc + 1\)" means the local time progress and "\(lc := 0\)" means the local clock initialization. \(current\_state\) is a variable in \(CS_{TH}\), which indicates the current state and the current local time.

For example, timed history variable in (Fig. 7) can be defined as a relation as follows:
Figure 7: Timed transition system for a timed history variable node

\[ TTS = \langle S_{TH}, s_0, C, A, R \rangle \]
\[ S_{TH} = \{(S_1, lc), (S_2, lc), (S_3, lc)\} \]
\[ s_0 = (S_1, 0) \]
\[ C = \{[a, b]|\text{Cond}_1, [0, a]|\text{Cond}_2, [0, a]|\text{Cond}_3, \text{Cond}_4\} \]
\[ A = \{\text{Assign}_1, \text{Assign}_2, \text{Assign}_3, \text{Assign}_4\} \]
\[ R = \{(S_1, (a, b)), \text{Cond}_1, \text{Assign}_1, S_2), ((S_2, [0, a]), \text{Cond}_3, \text{Assign}_3, S_1), ((S_2, [0, a])\text{Cond}_2, \text{Assign}_2, S_3), ((S_3, -), \text{Cond}_4, \text{Assign}_4, S_1)\} \]

From the definition of TTS above, a timed history variable node can be defined as a function \( f_{THV} \) from input values \( I_{THV} \) to output values \( O_{THV} \) as follows.

\[ f_{THV}(I_{THV}) = \sigma[I_{THV}/V_{THI}] = O_{THV} \]

Function \( f_{THV} \) generates an output \( O_{THV} \) whenever it gets inputs \( I_{THV} \) from other nodes in FOD. If no conditions are satisfied, then the value of \( O_{THV} \) in the previous scan cycle is preserved. However, although it gets no inputs \( I_{THV} \), the transition condition can be satisfied as the local time proceeds. In nuclear power plants system, however, this situation can be avoided. It is because system scan cycle time \( d \) is always much more smaller than the time \( a \) or \( b \) in timing constraint \([a, b]\). (i.e. \( d \) is 50ms and delay time \( a \) is 5sec.) Of course, we need to adjust that \( a \) or \( b \) are the multiple of \( d \).

**NuSCR Software System** NuSCR software system is defined as a tuple

\[ NSS = \langle S, S_0, R, d \rangle \] in which

- \( S \)
  - a set of system states
  - \( \sigma[V_{SE} \times CS_H \times CS_{TH}] \)
  - \( CS_H \) : a set of variables which indicate the current state of history nodes
  - \( CS_{TH} \) : a set of variables which indicate the current state and the current local time of time history variable nodes
- \( S_0 \) : initial state in \( S \)
- \( R \) : a set of transition relation \( S \times I \rightarrow S' \times O \)

11
• \(d\) : system scan cycle time in which the system get the changed valuation function \(\sigma\) periodically

5 Case Study: AMS Example

In this section, we introduce NuSCR software requirements specification for AMS (ATWS Mitigation System) in Kori NPP Unit 1 in Korea [19], [20]. We also introduce our supporting tool for NuSCR specification.

AMS Description

The AMS provides its protective action to mitigate the effects occurring followed by a failure of the reactor trip portion of the reactor trip system. It initiates a turbine trip and actuates AFWS (Auxiliary Feed Water System). There are two turbines and four AFWSs for each channel. The AMS also communicates some information with OP (Operational Panel), IPMS (In Plant Monitoring System), and Annunciators (i.e., alarms). The AMS consists of two identical channels containing Input/Output Module, Bistable Logic Module, and Coincidence Logic Module. The AMS and its related subsystems are described in (Fig. 8) and the description of its subsystems are as follows.

- **Input Validation Logic Module**: Input validation function converts the input parameters from raw input value to scaled value and validates the scaled input parameter.

- **Bistable Logic Module**: Bistable logic function determines if a trip condition exists based on measured parameters. Two types of bistable function are implemented, a fixed setpoint and a contact input. The fixed setpoint function compares a programmed constant with a digitized parameter to determine the trip state. The contact bistable function uses the state of the input to establish the trip state.

- **Coincidence Logic Module**: The coincidence logic function uses the bistable trip states generated internally from both channels to determine if a system initiation should occur. This function generates a turbine trip signal and a AFWS actuation signal whenever a coincidence of a low steam generator level trip or the manual trip has occurred.

- **Output Logic Module**: This logic shall obtain data from selected registers and send this information to the Annunciator, the In-Plant Monitoring System and the AMS OP.
NuSCR Specification for AMS  We produced NuSCR specification for a channel of AMS from [19] and [20]. (Fig. 9) depicts the gAMSOverview in NuSCR specifications, which is the root node of FOD. It shows the overall data flows of AMS systems. It has 17 inputs and 64 outputs. Inputs come from operation panel, external sensors, external hardware, and other channel. Outputs are for IPMS, communicators, operation panel, turbines, and AFWSs. It is identical to (Fig. 8). The AMS consists of 4 subgroup nodes, gInputValidationLogic, gBistableLogic, gCoincidenceLogic, and gOutputLogic.

The group node gBistableLogic in (Fig. 9) is decomposed into three group nodes gRisingTrip, gFallingTrip, and gDigitalTrip as depicted in (Fig. 4(a)), and the group node gBistableLogic is decomposed as (Fig. 4(b)). Finally three nodes in gRisingTrip are defined using SDT, FSM, and TTS respectively as (Fig. 1, 2, 3). As short of space, we introduce the part of AMS, gRisingTrip, as a typical NuSCR specification.

NuSCR Specification Supporting Tools  To be useful in developing practical systems, we provide a robust and well-engineered tool, NuEditor, for specifying the NuSCR specification. In NuEditor, simple properties s.t. completeness and consistency checking can be supported. Also it produces the adequate PVS inputs to verify the structural properties such as input/output completeness, consistency, and circular dependencies in NuSCR specification. It is based on our technique in [7]. We are now developing an automatic translating procedure from NuSCR specification into SMV inputs to verify further sophisticated properties. (Fig. 10) represents the NuEditor we are developing. With this tool, we are going to specify the whole system of RPS(Reactor Protection System), which is a core control process of nuclear power plant system, as a part of KNICS [8] project in Korea.

6 Conclusion and Future Work

Software safety is an important property for safety critical systems and formal requirements specification is known as a means to the safety in the early phase of software development process. Nowadays, in the area of nuclear power plants systems, the formal specification of software requirements is an urgent problem that needs to be solved right away with the replacement of existing
analog systems by digital systems composed of software process controllers.

In this paper, we introduce NuSCR, a formal software requirements specification method for digital protection system in nuclear power plants. NuSCR improves the readability and specifying ability by supplying different notations on the basis of the typical operation categories. The characteristics of the software process controller in nuclear power plants, s.t. periodic sequential processing and classifiable operations, makes this possible. We introduce the syntax and formal semantics of NuSCR to apply the recognized formal verification techniques to NuSCR specifications.

An ATWS mitigation system in Korean nuclear power plants is used as a case study to illustrate usefulness of our method. We also introduce the supporting tool, NuEditor, to be useful in developing practical systems. With this tool, we will specify the whole system of RPS(Reactor Protection System), which is a core control process of nuclear power plant system, as a part of KNICS [8] project in Korea. We are also developing an automatic translating procedure from NuSCR specification into SMV inputs to verify further sophisticated properties.

References


