NuSCR Manual
(ver. 1.0)

Dependable Software lab.
KAIST
Contents

- What is NuSCR?
- Background of NuSCR
- Components of NuSCR
- Variable naming rules
- FOD (Function Overview Diagram)
- Function Variable
- History Variable
- Timed History Variable
What is NuSCR?

- **Nu**clear + **SCR** (Software Cost Reduction)
- Fixed form language for describing requirements
- Suitable for software technology that receives input, performs control logic and gives output
- Suitable for nuclear energy field required technology
Background of NuSCR

◆ Expansion of the ACEL(Wolsong) method
◆ ACEL(Wolsong)
  ▪ Basic structure : FOD (Function Overview Diagram)
    ➢ Function : SDT (Structured Decision Table) function table
    ➢ History : State node + function
    ➢ Timing : Timing function
◆ NuSCR
  ▪ Basic structure : FOD
    ➢ Function : 개선된 SDT function table
    ➢ History : Automata
    ➢ Timing : Time Annotated Automata
Components of NuSCR

- Input variable
- Output variable
- Function variable
- History variable
- Timed history variable
- FOD (Function Overview Diagram)
Variable naming rules

- Add the corresponding prefix to each variable
  - $f_\_ :$ function variable
  - $h_\_ :$ history variable
  - $th_\_ :$ timed history variable
  - $i_\_ :$ input variable
  - $o_\_ :$ output variable
  - $k_\_ :$ predefined constant
  - $g_\_ :$ set of function variable, history variable, or timed history variable
FOD (Function Overview Diagram)

- A kind of DFD (Data Flow Diagram)
- Describes the relationships between the components of NuSCR
- Display each component with a node
- Display relationships between nodes with one-way arrows
- Use group nodes when composed in classes
- Each node name follows the variable naming rule
Elements represented in FOD

- Input node, Output node
- Group node
- Function node
- History node
- Timed history node
- Data Flow or Transition
Example of FOD (1/3)
Example of FOD (2/3)

g_BP(detailed) + External Input/Output
Example of FOD (3/3)

\[ g_{BP} > g_{VAR\_OVER\_PWR} \]
Function Variable

- Used to describe the system’s functional behavior
- Defined with SDT (Structured Decision Table)
  - SDT is a type of Condition/Action table
  - Once the condition is satisfied, the action is performed
  - Familiar table style for the engineer
**SDT (Structured Decision Table)**

- **Condition**
  - Complex condition composed of function variable inputs
  - ie) \( k_{X\_MIN} \leq f_X \leq k_{X\_MAX} \)

- **Action**
  - Assignments for function variables
  - ie) \( f_{X\_Valid} := 0 \)
Examples of SDT

<table>
<thead>
<tr>
<th>Conditions</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>k_X_MIN &lt;= f_X &lt;= k_X_MAX</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actions</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_X_Valid := 0</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>f_X_Valid := 1</td>
<td></td>
<td>O</td>
</tr>
</tbody>
</table>

- SDT defines the function Variable f_X_Valid
- Meaning
  - If f_X is greater than or equal to k_X_MIN, and less than or equal to k_X_MAX (condition),
  - Assign 0 to f_X_Valid (action)
Example of SDT from RPS items

Example of function variables defined through SDT

Structured Decision Table:

<table>
<thead>
<tr>
<th>Conditions</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_LO_SG1_LEVEL_Val_OUT &gt; k_LO_SG1_LEVEL_PV...</td>
<td>T</td>
<td>-</td>
<td>F</td>
</tr>
<tr>
<td>f_LO_SG1_LEVEL_Val_OUT &lt; k_LO_SG1_LEVEL_PV...</td>
<td>-</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

Action

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_LO_SG1_LEVEL_PV_Err := true</td>
<td>0</td>
<td>O</td>
<td>0</td>
</tr>
<tr>
<td>f_LO_SG1_LEVEL_PV_Err := false</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
History Variable

- Used to describe system’s condition based action
- Defined with a FSM (Finite State Machine)
  - Components of FSM
    - Finite number of states
    - Transitions between states
FSM (Finite State Machine)

- **State**
  - Express each of the system’s states
  - ie) A switch has two states: On and Off

- **Transition**
  - Represents the changes between states
  - Expressed with arrows
  - Each transition has a label
  - label form $\rightarrow$ Conditions/Actions
Example of FSM (Finite State Machine)

- **FSM that defines the history variable h_X_OB_Sta**
- **Meaning**
  - In the initial state NOT_OB_STATE
  - If the conditions $f_{X\_OB\_Perm} = 1$ and $f_{X\_OB\_Ini} = 1$ are satisfied (condition)
  - Assign the value 1 to $h_{X\_OB\_Sta}$ (action)
  - Move to the OB_State (transition)
Example of FSM from RPS items

Example of history variables defined through FSM

- **Condition**: \( f_{HB\_In\_t0} < f_{HB\_In} \) & \( h_{HB\_Err\_Cnt} < 1 \)
- **Action**: \( h_{HB\_Err\_Cnt} := 0 \)
Timed History Variable

- Used to describe system’s time related actions
- Defined with TTS (Timed Transition System)
  - TTS is an extension of FSM
  - Time Annotated Automata
  - Adds a time restriction to FSM’s transition condition
  - Attaches a time restriction in the form of $[a,b]$ in front of the condition
TTS (Timed Transition System)

- **State**
  - Describes the systems’ different states

- **Transition**
  - Represents the changes between states
  - Expressed with arrows
  - Every transition has a label
  - label format $\rightarrow [\text{Time}_1, \text{Time}_2] \text{Conditions/Actions}$
  - ie) $[1,4]\text{condition}=0/\text{action}:=1$
    - If the condition=0 is maintained for a term of 1~4 hours, assign action=1 and change state
Example of TTS (Timed Transition System)

- **TTS that defines a part of Timed History Variable** `th_X_Trip`

- **Meaning**
  - In Waiting state
  - For `k_Trip_Delay` hours (Time Limit)
  - If `f_X >= k_X_Trip_SetPoint` and `h_X_OB_Sta = 0` conditions are satisfied and maintained (condition)
  - Assign `th_X_Trip` the value 0 (action)
  - Move to the Trip_By_Logic state (transition)
Example of TTS from RPS items

- Example of Timed History Variable defined through TTS

- **Time duration**: \([k\_HI\_LOG\_POWER\_Trip\_Dly, k\_HI\_LOG\_POWER\_Trip\_Dly]\)
- **Condition**: \(f\_HI\_LOG\_POWER\_Val\_Out > k\_HI\_LOG\_PWR\_Trip\_Set \)
- **Action**: \(th\_HI\_LOG\_PWR\_Trip\_Logic := true\)