A Domain-Specific Safety Analysis for Digital Nuclear Plant Protection Systems

Sanghyun Yoon
Dependable Software Laboratory
Konkuk University, Korea

June 28, 2011
Contents

• Introduction

• Background
  – Software Fault Tree Analysis
  – A Formal Software Requirement Specification method, NuSCR

• Our suggestion: NuFTA
  – Overview of NuFTA
  – Software fault tree templates for NuSCR nodes
  – Experimental Result

• Conclusion & Future Work
Introduction(1)

• Failures of safety-critical systems incur catastrophic disaster
  – The systems require rigorous quality demonstration.

• Safety analysis tries to assure the systems’ safety through performing various safety analysis techniques
  – FTA (Fault Tree Analysis), FMEA (Failure Mode and Effect Analysis), HAZOP (Hazard and Operability study).

• Safety experts apply the techniques manually
  – Quality and correctness of the analysis result totally depends on the knowledge and experience of the experts.
Introduction(2)

- Many safety analysis techniques focus on mechanical generation of software fault tree.

- If we restrict the application domain of safety analysis into some critical failures, we can use the safety analysis techniques more efficiently.

- Our target domain was KNICS (Korea Nuclear Instrumentation and Control System) RPS (Reactor Protection System).

- Prototype version of KNICS RPS is specified with NuSCR.

- We propose a CASE tool, NuFTA
  - NuFTA is a CASE tool for digital nuclear RPS.
  - NuFTA generates software fault tree mechanically from an NuSCR specification.
BACKGROUND
Software Fault Tree Analysis

- **Software Fault Tree Analysis (SFTA)**
  - Target of SFTA is software of a system.
  - Deductive and top-down method of analyzing system.
  - Identifying all of the associated elements using boolean gate that could cause top event (failure) to occur.

- **Minimal cut-set**
  - A basic set of events that can cause failure.
  - Safety experts use minimal cut-set to obtain an estimate of reliability for complex fault tree.

Subsystem A =
\[(1 \mid 2) \mid (3 \& 4 \& 5) \mid (6 \mid (7 \& 8))\]

<Minimal cut-set of subsystem A>

<A fault tree for subsystem A>
A Formal Software Requirement Specification method
- NuSCR(1)

- Extended SCR (Software Cost Reduction, Heninger, 1980) for RPS

- Sequential System.

- An FOD(Function Overview Diagram) is composed of variable nodes.

- Variable nodes
  - Function variable node(SDT), prefix : f
  - History variable node(FSM), prefix : h
  - Timed-history variable node(TTS), prefix : th
A Formal Software Requirement Specification method - NuSCR(2)

- Function variable node is defined with structured decision table (SDT).
- SDT is composed of condition statements and action statements.

Structured Decision Table:

<table>
<thead>
<tr>
<th>Conditions</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>th_VAR_OVER_PWR_Trip_Loic = true &amp; f_VAR_OVER_PWR_Op_Byp_Init = false</code></td>
<td>T</td>
<td>-</td>
<td>F</td>
</tr>
<tr>
<td>`f_Mod_Err = true</td>
<td>f_VAR_OVER_PWR_Chan_Err = true</td>
<td>f_VAR_OVER_PWR_PV_Err = true`</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>f_VAR_OVER_PWR_Trip_Out := true</code></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td><code>f_VAR_OVER_PWR_Trip_Out := false</code></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

<A definition of function variable node (Structured Decision Table)>
A Formal Software Requirement Specification method - NuSCR(3)

Timed-history variable node (Timed Transition System)
Overview of NuFTA

• Purpose
  – Mechanically generates a software fault tree for analysts.
  – Root node of SFT: trip/pre-trip(shut-down) signal
  – Analysis result: graphical fault tree, logical expression

• Software fault tree constructing process using NuFTA
  1) Analyst selects a node generating shutdown signal in NuSRS (NuSCR supporting tool).
  2) The NuFTA analyzes backwardly causes of the signal throughout all connected nodes in an FOD.
  3) Using fault tree templates for NuSCR nodes, the NuFTA constructs a software fault tree for the node.
  4) The NuFTA produces a logical expression(minimal cut-set) representing the generated software fault tree.
Software fault tree templates for NuSCR nodes(1)


- We modified templates and used for developing NuFTA.

- NuFTA uses software fault tree templates for analyzing variable nodes of NuSCR specifications.

- For analyzing NuSCR nodes, the templates classifies
  - Relational operator of action/assign statement
  - Definition of right hand side of action/assign statement
Software fault tree templates for NuSCR nodes (2)

- This part of SDT template classifies relational operator of action statement.

\[
f_{\text{VAR\_OVER\_Trip\_Out}} := \text{true}
\]

\text{Structured Decision Table:}

\begin{tabular}{|c|c|c|c|}
\hline
\text{Conditions} & 1 & 2 & 3 \\
\hline
\text{th\_VAR\_OVER\_FUR\_Trip\_logic} = \text{true} & \text{false} & \text{false} & \text{false} \\
\hline
\text{f\_Mod\_Err} = \text{true} & \text{false} & \text{true} & \text{true} \\
\text{f\_VAR\_OVER\_FUR\_Chm\_Err} = \text{true} & \text{false} & \text{true} & \text{true} \\
\text{f\_VAR\_OVER\_FUR\_Fur\_Err} = \text{true} & \text{false} & \text{true} & \text{true} \\
\hline
\text{Action} & 1 & 2 & 3 \\
\hline
\text{f\_VAR\_OVER\_FUR\_Trip\_Out} := \text{true} & 0 & 0 & 0 \\
\text{f\_VAR\_OVER\_FUR\_Trip\_Out} := \text{false} & 0 & 0 & 0 \\
\hline
\end{tabular}
Software fault tree templates for NuSCR nodes(3)

- This part of SDT template classifies definition of RHS of action statement.

- function variable node = constant
  - e.g. $f_X = 1$

- function variable node
  = other variable node + constant
  - e.g. $f_X = th_{Trip\_Logic} + 1$
  - NuFTA additionally attaches a sub-tree for output value of $th_{Trip\_Logic}$

- function variable node
  = function variable node + constant
  - RHS has output value of previous cycle
  - e.g. $f_X = f_X + 1$
  - NuFTA additionally attaches a sub-tree for output value of $f_X$ on previous cycle.

<A template for SDT(2)>
Annotated automata

- History and timed-history variable are defined with automata
  - Output values of automata are not specified on states
  - We need to specify output values on states for algorithmic analysis.

- Our suggestion: Annotated automata
  - Unfolded automata whose states specified own output value.
  - NuFTA unfolds automata then analyze the annotated automata.
Software fault tree templates for NuSCR nodes(4)

- A template for TTS(1)
Software fault tree templates for NuSCR nodes(5)

- A template for TTS(2)
Software fault tree templates for NuSCR nodes (6)

- A template for FSM(3) – enter the state
Software fault tree templates for NuSCR nodes

- A template for FSM(4) – remain at the state

![Diagram showing fault tree templates for NuSCR nodes](image)

Condition of self-cycling transition is satisfied
A screen dump of NuFTA

A full generated software fault tree using NuFTA
Experimental Result(1)

- We used a prototype version of requirement specification of KNICS RPS.
Experimental Result(2)

<table>
<thead>
<tr>
<th>Name of FOD</th>
<th>Range of a process variable</th>
<th>Analysis time of $\text{trip_out}(\text{ms})$</th>
<th>Analysis time of $\text{pretrip_out}(\text{ms})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_\text{VAR_OVER_PWR}$</td>
<td>0~100</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$g_\text{LO_SG1_LEVEL}$</td>
<td>0~100</td>
<td>138</td>
<td>109</td>
</tr>
<tr>
<td>$g_\text{HI_LOG_POWER}$</td>
<td>0~100</td>
<td>92</td>
<td>142</td>
</tr>
<tr>
<td>$g_\text{LO_PZR_PRESS}$</td>
<td>0~100</td>
<td>205</td>
<td>197</td>
</tr>
<tr>
<td>$g_\text{SG1_LO_FLOW}$</td>
<td>0~100</td>
<td>111</td>
<td>108</td>
</tr>
<tr>
<td>$g_\text{HI_LOCAL_POWER}$</td>
<td>0~2</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

- NuFTA constructed SFT from FODs, except the most complex FOD.
- Cause of this problem: state explosion problem
  - Optimization of source code and data structure is required.
Conclusion & Future Work

• Conclusion
  – NuFTA is a CASE tool supporting software fault tree analysis for analysts.
  – We restricted application domain of safety analysis into specific type of critical failure, ‘shut down’.
  – We automated large part of safety analysis.

• Future work
  – Optimization of code and data structure
  – Definition of semantics for time constraints