Formal Modeling and Verification of Safety-Critical Software implemented in PLC

JUNBEOM YOO

KONKUK University, Korea
jbyoo@konkuk.ac.kr  http://dslab.konkuk.ac.kr
Other Authors

Sungdeok Cha  
- Professor in Korea University

Enukyoung Jee  
- PhD Candidate in KAIST
Contents

• Introduction
  – Safety-Critical Software in Nuclear Power Plants
  – Software Development Process (Existing vs. Proposed)

• Software Development Process for NPPs
  – Development Process
  – Verification Process
  – Safety Analysis Process

• Conclusion and Future Work
Introduction

1. Safety-Critical Software in Nuclear Power Plants
2. Software Development Process (Existing vs. Proposed)
Safety-Critical Software in Nuclear Power Plants

- RPS (Reactor Protection System)
- ESF-CCS (Engineering Safety Features Component Control System)
Existing Software Development Process

- For Most NPPs in Korea (e.g. Wolsung NPP)
**Proposed Software Development Process**

- For KNICS RPS for APR-1400 [1] (http://www.knics.re.kr)
  - APR-1400: Next generation nuclear reactor being developed in Korea
Software Development Process for NPPs

1. Development Process
2. Verification Process
3. Safety-Analysis Process
Development Process

1. Formal Requirements Specification
2. Automatic Design Synthesis
1. Formal Requirements Specification

- NuSCR [3]
  - Formal requirements specification language
  - Customized SCR [2] for nuclear applications
    - Listened to opinions offered by domain experts
  
- 4 constructs
  - SDT (Structured Decision Table)
  - FSM (Finite State Machine)
  - TTS (Timed Transition System)
  - FOD (Function Overview Diagram)

```
<table>
<thead>
<tr>
<th>Conditions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T.&lt;=X.MIN &lt;= T.X &lt;= T.X.MAX</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T.X.Valid := 0</td>
<td>X</td>
</tr>
<tr>
<td>T.X.Valid := 1</td>
<td>X</td>
</tr>
</tbody>
</table>
```

SDT

TTS / FSM
1. Formal Requirements Specification

- **NuSRS (ver 2.0)**
  - CASE tool supporting
    - NuSCR specification
    - Self-Checking (on-going)
    - SMV program translation (NuSCR $\rightarrow$ SMV)
    - SMV verification (CTL Model Checking)

- **Case Study**
    (by NuSRS 1.0)
  - KNICS-RPS-SVR131-01, Rev.00, 2005.
    (by NuSRS 2.0)
2. Automatic Design Synthesis

- **NuSCRtoFBD Synthesis Procedure [8]**
  - Synthesizes FBD programs from NuSCR specification automatically
    - More than twice FBD blocks than manually coded and optimized ones
  - Unused in the project, because unable to develop CASE tools in advance
  - However, can be used as a baseline for FBD programming in design phase

- **NuSCRtoFBD (ver 1.0)**
  - CASE tool supporting
    - Automatic FBD synthesis from NuSCR
      - Reads NuSCR specification in XML format
      - Stores FBD programs in standard XML format (on-going)
    - Algorithm is being optimized
NuSCRtoFBD (ver. 1.0)
- Synthesized from KNICS RPS BP SRS (KNICS-RPS-SVR131-01, Rev.00, 2005)
Verification Process

1. Model Checking Requirements
2. Model Checking Design
3. Equivalence Checking Designs
1. Model Checking Requirements

- Formal verification for requirements specification
  - Target: NuSCR formal specification
  - Tool: Cadence SMV [5]
  - Technique: CTL model checking

- NuSRS (ver. 2.0)
  - Automatic translation from NuSCR into SMV programs [10]
  - Seamless execution of SMV

- Case Study
  - KNICS-RPS-SVR131-01, Rev.00, 2005
  - Found 157 errors (25 critical)
FBD Verification using
- SMV model checking & VIS Equivalence checking
2. Model Checking Design

- Formal verification for design specification
  - Target: FBD program
  - Tool: Cadence SMV [5]
  - Technique: LTL model checking

- FBD Verifier (ver. 1.0 / 2.0)
  - Automatic translation from FBD programs into Verilog programs [11]
  - Seamless execution of SMV

- Case Study
  - KNICS-RPS-SDS231, Rev.01, 2006
  - Found 60 errors (13 critical)
1. Read FBD programs in XML format

2. Translate into Verilog program

3. Perform SMV model checking

4. Analyze verification result

Engineering Tools by PLC vendors

Countercexample viewer in FBD Verifier 1.0

Cadence SMV

FBD Verifier 1.0
3. Equivalence Checking Designs

- Formal verification for design specifications
  - Target: Two FBD programs
  - Tool: VIS Verification System [4]
  - Technique: Equivalence checking, Simulation

- VIS Analyzer (ver. 1.0)
  - Seamless execution of VIS (VIS has no GUI)
  - Visualization of VIS’s process and verification results [12]
  - Unused in the project, because unable to develop CASE tools in advance

- Case Study
  - KNICS-RPS-SDS101, Rev.00, 2005
  - No official result

<table>
<thead>
<tr>
<th>Trip Logic</th>
<th>Error Type</th>
<th>Compared FBD (Num. of Errors)</th>
<th>Original FBD (Num. of Errors)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed Set-Point Rising Trip</td>
<td>Syntactic</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>without Operating Bypass</td>
<td>Logical</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manual Reset Variable Set-Point</td>
<td>Syntactic</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Trip without Operating Bypass</td>
<td>Logical</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>
Execute VIS equivalence checking

Execute VIS simulation
### VIS Analyzer (ver. 1.0)
- Visualized and reorganized result - counterexample

<table>
<thead>
<tr>
<th># state</th>
<th>input</th>
<th>File1 Output</th>
<th>File2 Output</th>
<th>File1 State</th>
<th>File2 State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Initial</td>
<td>Initial</td>
<td>Initial</td>
<td>S11 T0</td>
<td>S01 T0</td>
</tr>
<tr>
<td>1</td>
<td>61</td>
<td>1</td>
<td>1</td>
<td>S11 T1</td>
<td>S11 T1</td>
</tr>
<tr>
<td>2</td>
<td>61</td>
<td>1</td>
<td>1</td>
<td>S11 T2</td>
<td>S11 T2</td>
</tr>
<tr>
<td>3</td>
<td>61</td>
<td>1</td>
<td>1</td>
<td>S11 T3</td>
<td>S11 T3</td>
</tr>
<tr>
<td>4</td>
<td>61</td>
<td>1</td>
<td>1</td>
<td>S11 T4</td>
<td>S11 T4</td>
</tr>
<tr>
<td>5</td>
<td>61</td>
<td>1</td>
<td>1</td>
<td>S11 T5</td>
<td>S11 T5</td>
</tr>
<tr>
<td>6</td>
<td>61</td>
<td>0</td>
<td>0</td>
<td>S00 T5</td>
<td>S20 T5</td>
</tr>
<tr>
<td>7</td>
<td>52</td>
<td>0</td>
<td>0</td>
<td>S00 T0</td>
<td>S20 T0</td>
</tr>
<tr>
<td>8</td>
<td>52</td>
<td>1</td>
<td>0</td>
<td>Null</td>
<td>Null</td>
</tr>
</tbody>
</table>
Safety Analysis Process

1. Fault Tree Analysis for Requirements
2. Fault Tree Analysis for Design
1. Fault Tree Analysis for Requirements
2. Fault Tree Analysis for Design

- Fault Tree Analysis
  - performed manually
  - Totally depends on analyst’s experience and ability

- We provided FTA templates for NuSCR [13] and FBD [15]

FTA templates for FBDs
Conclusion and Future Work
Conclusion

- We proposed software development processes using formal methods
  - Target: KNICS RPS for APR-1400

- Development process
  - NuSCR formal requirements specification
  - Automatic FBD design synthesis

- Verification process
  - Model checking NuSCR requirements
  - Model checking FBD design
  - Equivalence checking FBD designs

- Safety analysis process
  - FTA templates for NuSCR requirements
  - FTA templates for FBD programs

- Case Study
  - KNICS-RPS-SVR131-01, Rev.00, 2005
  - KNICS-RPS-SDS231, Rev.01, 2006
Future Work

1. Integrated Tool-set

2. Tool Enhancement
   - Self-checking: completeness & consistency (NuSRS)
   - Synchronous Verilog issue in model checking FBD programs using SMV (FBD Verifier)
   - Optimization of FBD synthesis algorithm (NuSCRtoFBD)
   - Add other functions to VIS Analyzer (VIS Analyzer)

3. Traceability Analysis
   - From requirements to design
   - From requirements’ FTA to design’s FTA

4. FBD Testing
   - Measures (coverage criteria)
   - Testing tool support

5. Application to Other Domains
References


Relay-based Analog System

PLC-based Digital System
NuSRS 2.0
- Full specification for KNICS RPS BP SRS (KNICS-RPS-SVR131-01, Rev.00, 2005)
<table>
<thead>
<tr>
<th></th>
<th>BP</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target subsystems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Information</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#pages of natural lang. spec.</td>
<td>60</td>
<td>46</td>
</tr>
<tr>
<td>#group nodes of NuSCR spec.</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>#nodes of NuSCR spec.</td>
<td>81</td>
<td>111</td>
</tr>
<tr>
<td>Verification Information</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#properties</td>
<td>124</td>
<td>207</td>
</tr>
<tr>
<td>Detected Errors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incorrect specification</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Omission</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>Logic check required</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Incorrect formal specification</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Syntax error of formal specification</td>
<td>83</td>
<td>45</td>
</tr>
<tr>
<td>Total #errors</td>
<td>104</td>
<td>53</td>
</tr>
<tr>
<td>Primary #errors</td>
<td>19</td>
<td>6</td>
</tr>
</tbody>
</table>

SMV Verification Result
- KNICS RPS BP & CP SRS (KNICS-RPS-SVR131-01, Rev.00, 2005)
<table>
<thead>
<tr>
<th>Target subsystems</th>
<th>BP</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Information</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#pages of natural lang. spec.</td>
<td>190</td>
<td>163</td>
</tr>
<tr>
<td>#function blocks</td>
<td>1,335</td>
<td>1,623</td>
</tr>
<tr>
<td>#variables</td>
<td>1,038</td>
<td>820</td>
</tr>
<tr>
<td>#lines of Verilog model</td>
<td>7,862</td>
<td>3,085</td>
</tr>
<tr>
<td>Verification Information</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#properties</td>
<td>216</td>
<td>83</td>
</tr>
<tr>
<td>Detected Errors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incorrect logic</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>Omission</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Incorrect in certain condition</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Incorrect FBD</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>Incorrect SDS</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>Total #errors</td>
<td>47</td>
<td>13</td>
</tr>
<tr>
<td>Distinct #errors</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

SMV Verification Result
- KNICS RPS BP & CP SDS (KNICS-RPS-SDS231, Rev.01, 2006)