Supervisory Logic & Fault Management in MATLAB/Simulink

Nordic MATLAB EXPO 2016

Hossein Mousavi
Application Engineering Manager
MathWorks Nordic
## Challenges of Product Development

### Table 2: Top Six Challenges of Mechatronic Product Development

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulty finding and hiring experienced system engineers / lack of cross-functional knowledge</td>
<td>50%</td>
</tr>
<tr>
<td>Early identification of system level problems</td>
<td>45%</td>
</tr>
<tr>
<td>Ensuring all design requirements are met in the final system</td>
<td>40%</td>
</tr>
<tr>
<td>Difficulty predicting / modeling system product behavior until physical prototypes exist</td>
<td>32%</td>
</tr>
<tr>
<td>Difficulty implementing an integrated product development solution for all disciplines involved in mechatronic product development</td>
<td>28%</td>
</tr>
<tr>
<td>Inability to understand the impact a design change will have across disciplines</td>
<td>18%</td>
</tr>
</tbody>
</table>

Data source:
*System Design: New Product Development for Mechatronics*
*January 2008, Aberdeen Group, Survey size: 160 enterprises*
Motivation
Multi-Domain Engineering in ONE environment

- Mathematical algorithms
- **Logic** (modes, supervisory, fault)
- Physical components (plant)
- Data driven (system identification)
- Code (legacy, MATLAB)
Multi-Domain Engineering in ONE environment

Controller

Plant

DESIGN

Environmental Models

Mechanical  Electrical

Control Algorithms

Supervisory Logic
Example: Control Logic for a Fan Cooling System

- Device *generates heat* when it is switched on
Example: Control Logic for a Fan Cooling System

- Device **generates heat** when it is switched ON
- **Temperature** of device rises when it is ON
- Cooling fan **switches ON** when temperature exceeds threshold
- Cooling fan has **modes** Off, Low and High to regulate temperature
Example: Control Logic for a Fan Cooling System

- Device **generates heat** when it is switched ON
- **Temperature** of device **rises** when it is ON
- Cooling fan **switches ON** when temperature exceeds threshold
- Cooling fan has **modes** Off, Low and High to regulate temperature
- 2 fans for redundancy
DEMO: Supervisory Logic – High level
Challenges to Designing Control Logic

- Modes of operation are difficult to represent with traditional approaches
- Conditional logic can be very complex
- Interaction between components is hard to visualize and debug
Solution: State machines and Flow Charts in Stateflow

- State machines are regularly used to represent mode logic
- Flow charts are used to model processes that contain conditions
- In Stateflow you can visualize the behavior of logic by animating it during simulation
Other Approaches for Modeling Control Logic

Simulink

```c
void a_step(void) {
    if (a_DWBRK.is_active_on_a == 0) {
        a_DWBRK.is_active_on_a = 10;
        if (a_DWBRK.is_on_b == 0) {
            a_DWBRK.is_on_b = 10;
            a_DWBRK.is_on_b = passive;
            a_B.y.LO_mode = Passive;
        }
    } else {
        switch (a_DWBRK.is_on_b) {
            case 10:
                a_DWBRK.is_on_b = 0;
                break;
            case 0:
                a_DWBRK.is_on_b = 10;
                a_B.y.LO_mode = Standby;
                break;
        }
    }
}
```
Other Approaches for Modeling Control Logic

Stateflow

Code

>1000 lines
How do we actually create the state chart?
Questions:

- We want to start test and verify our design. What is the most common way to do this?

Answer:

- Use the play-button for simulation!
- Test Automation through MATLAB scripting
Testing Using Simulation

- Input Test Vector
- Analyze results!

Model Used for Production Code Generation

1. Design Verification
- Textual Requirements
- Executable Specification
- Model Used for Production Code Generation

2. Code Verification
- C Source Code
- Executable Object Code
Questions:

- How do we know if enough testing performed on our Control Logic?
- All parts tested?

Answer:

- Coverage analysis!
Model Coverage Report

- Coverage metrics identifies untested portions of your model

Coverage Report sbr

Tests

Test 1
Start Execution: 27-Feb-2008 13:36:21
End Execution: 27-Feb-2008 13:36:21

Summary

<table>
<thead>
<tr>
<th>Model Hierarchy/Complexity</th>
<th>Test 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. sb</td>
<td>D1</td>
</tr>
<tr>
<td>2. ... Inputs</td>
<td>58%</td>
</tr>
<tr>
<td>3. ... Antenna Tests</td>
<td>50%</td>
</tr>
<tr>
<td>4. ... Outputs: Assertion</td>
<td>100%</td>
</tr>
<tr>
<td>5. ... Verification Subsys</td>
<td>100%</td>
</tr>
<tr>
<td>6. ... SBR Logic</td>
<td>100%</td>
</tr>
<tr>
<td>7. ... SBR</td>
<td>100%</td>
</tr>
<tr>
<td>8. ... SP SBR</td>
<td>100%</td>
</tr>
<tr>
<td>9. ... SP KEY ON</td>
<td>100%</td>
</tr>
<tr>
<td>10. ... SP SB UNFASTEN</td>
<td>75%</td>
</tr>
<tr>
<td>11. ... SP HIGH SPEED</td>
<td>100%</td>
</tr>
<tr>
<td>12. ... SB Logic</td>
<td>100%</td>
</tr>
<tr>
<td>13. ... SBB</td>
<td>100%</td>
</tr>
</tbody>
</table>

Summary

<table>
<thead>
<tr>
<th>Test 1</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>100%</td>
</tr>
<tr>
<td>1. sb</td>
<td>100%</td>
</tr>
<tr>
<td>2. ... Inputs</td>
<td>58%</td>
</tr>
<tr>
<td>3. ... Antenna Tests</td>
<td>50%</td>
</tr>
<tr>
<td>4. ... Outputs: Assertion</td>
<td>100%</td>
</tr>
<tr>
<td>5. ... Verification Subsys</td>
<td>100%</td>
</tr>
<tr>
<td>6. ... SBR Logic</td>
<td>100%</td>
</tr>
<tr>
<td>7. ... SBR</td>
<td>100%</td>
</tr>
<tr>
<td>8. ... SP SBR</td>
<td>100%</td>
</tr>
<tr>
<td>9. ... SP KEY ON</td>
<td>100%</td>
</tr>
<tr>
<td>10. ... SP SB UNFASTEN</td>
<td>75%</td>
</tr>
<tr>
<td>11. ... SP HIGH SPEED</td>
<td>100%</td>
</tr>
<tr>
<td>12. ... SB Logic</td>
<td>100%</td>
</tr>
</tbody>
</table>

- Decision
- Condition
- MC/DC
- Lookup table
- Signal range
Questions:

- What if not high enough coverage with current test cases?
- What if not reaching specific scenario (hard-to-reach)?

Answer:

- Additional testing requirements -> Additional test cases
- Automatic test case generation!
Automatic Test Generation

- To reach full coverage or user-defined objectives (specific scenarios)

- Test vectors generated (structural) can be used for equivalence test
  - Software-in-the-Loop (SIL)
  - Processor-in-the-Loop (PIL)