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

Controller → Plant

Controller → Plant → Controller → Plant
Multi-Domain Engineering in ONE environment

- Mathematical algorithms
- **Logic (modes, supervisory, fault)**
- Physical components (plant)
- Data driven (system identification)
- Code (legacy, MATLAB)

![Diagram of a control system with labels: Controller, Plant, u, +, y, s1, s2, s3, Actuators, Sensors.](image)
Multi-Domain Engineering in ONE environment
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

void a_step(void)
{
if (a_DWICK.is_active_v1_a == 0) {
    a_DWICK.is_active_v1_a = 1;
}
if (a_DWICK.is_active_v2_b == 1) {
    a_DWICK.is_active_v2_b = 0;
}
if (a_DWICK.is_active_v3_c == 0) {
    a_DWICK.is_active_v3_c = 1;
}

else {
    switch (a_DWICK.is_active_v3_c) {
    case a_DWICK.is_active_v3_c;
        if (!((a_DWICK.is_active_v2_b == 0) && (a_DWICK.is_active_v1_a == 1))) {
            a_DWICK.is_active_v1_a = 1;
        }
        else {
            a_DWICK.is_active_v2_b = 1;
        }
        break;
    case a_DWICK.is_active_v3_c:
        if (!((a_DWICK.is_active_v2_b == 0) && (a_DWICK.is_active_v1_a == 1))) {
            a_DWICK.is_active_v1_a = 1;
        }
        else {
            a_DWICK.is_active_v2_b = 1;
        }
        break;
    case a_DWICK.is_active_v3_c:
        if (!((a_DWICK.is_active_v2_b == 0) && (a_DWICK.is_active_v1_a == 1))) {
            a_DWICK.is_active_v1_a = 1;
        }
        else {
            a_DWICK.is_active_v2_b = 1;
        }
        break;
    case a_DWICK.is_active_v3_c:
        if (!((a_DWICK.is_active_v2_b == 0) && (a_DWICK.is_active_v1_a == 1))) {
            a_DWICK.is_active_v1_a = 1;
        }
        else {
            a_DWICK.is_active_v2_b = 1;
        }
        break;
    case a_DWICK.is_active_v3_c:
        if (!((a_DWICK.is_active_v2_b == 0) && (a_DWICK.is_active_v1_a == 1))) {
            a_DWICK.is_active_v1_a = 1;
        }
        else {
            a_DWICK.is_active_v2_b = 1;
        }
        break;
    default:
        a_DWICK.is_active_v1_a = a_DWICK.is_active_v1_a;
        break;
    }
}

~50 lines
Other Approaches for Modeling Control Logic

Stateflow

Code

>1000 lines
DEMO: Simple State Machine (subpart)

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!
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 for sbr

Tests

Test 1

Started Execution: 27-Feb-2008 13:36:21
Ended Execution: 27-Feb-2008 13:36:21

Summary

Model Hierarchy/Complexity:

1. sbr
   2. ... inputs
   3. ... Analysis Tests
   4. ... Output Assertions
   5. ... Verification Subsystem
   6. ... SBR Logic
   7. ... SBR
   8. ... SP SBR
   9. ... SP KEY ON
   10. ... SP SB UNFASTEN
   11. ... SP HIGH SPEED
   12. ... SBR Logic
   13. ... SBR

Test 1

<table>
<thead>
<tr>
<th>CI</th>
<th>Test</th>
<th>Model Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></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)