Continuous

Test & Verification

Model & Code
System & Component
Dynamic testing & Static analysis

Productivity
QUALITY
Needs verification!
continuous
Multi-Mode Hybrid Electric Vehicle with Model Based Design

Multi-Mode Hybrid Electric Vehicle

Define Requirements

System-Level Specification

Subsystem Design

Subsystem Implementation

Simulink Plant Model

Production C-Code

Controller Vehicle

Battery

Engine

Energy Management

Simulink Integration & Test

Subsystem Integration & Test

Failure

Complete Integration & Test

System-Level Integration & Test

MATLAB EXPO 2017
Continuous Test and Verification Framework

- **Repeat**: Reproduce the failure at the simulation level
- **Slice**: Isolate the problematic behavior
- **Fix**: Fix and Perform Unit Testing
- **Check**: Check for further design errors
- **Test**: Test Systematically and Test Completely

Productivity + Quality
needs verification!
continuous
Reproduce the Failure in Simulation

Confirm the problem In the Lab/Desktop Simulation
Failure Report

Reproduce Failure

Drive cycle

MATLAB EXPO 2017
Simulation Environment

Controller

Plant
Modeling the Test

Drive cycle

Controller

Plant

MATLAB EXPO 2017
Modeling the Test

Safety Property

Engine RPM must remain within operating bounds limits

Symbols

Input

1. EngSignals
2. BattSignals
3. GenSignals
4. VehSignals
5. ClutchSignals
6. MotSignals

Step

Assessment

Test Assessment
Battery State of Charge

Initial state of charge?

Sweep from 100% to 50%
New Test Suite 1

- Test Suite
- TAGS
- DESCRIPTION
- REQUIREMENTS
- CALLBACKS
- COVERAGE SETTINGS

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>New Test Suite 1</td>
</tr>
<tr>
<td>Location</td>
<td>C:\work\Mab הדין</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>HEVM_Test » New Test Suite 1</td>
</tr>
<tr>
<td>Enabled</td>
<td>True</td>
</tr>
<tr>
<td>Record Coverage</td>
<td>False</td>
</tr>
<tr>
<td>Tags</td>
<td>type comma or spc</td>
</tr>
</tbody>
</table>
Model Slicing: Isolate the problematic behavior

Simulation Scenario → Isolate the problem → Slice → Fix → Check → Test → Repeat
Challenge of Understanding Behavior

- Complexity of Plant
Challenge of Understanding Behavior

- Complexity of Plant
- Complexity of Controller
Challenge of Understanding Behavior

- Complexity of Plant
- Complexity of Controller
- Complexity of Dynamics

Turning backwards
Model Slicing
Complete Model Slicer Workflow

- Identify Interest
- Slice Model
- Create Slice Model
- Iterate
- Incorporate Changes
- MBD Work:
  - Simulate/Debug
  - Formal analysis
  - Update
- Updated Slice Model
- Highlighted Model
- Original Design Model
Isolating Troubling Behavior with Model Slicer

Simulation Scenario → Model Slicer → Area of Interest

Highlight of Relevant Parts

Controller → Plant

Stateflow Highlighting R2016b

Identify Interest → Original Design Model → Incorporate Changes

Identify Interest → Highlighted Model → Iterate

Create Slice Model → Slice Model → MBD Work

MATLAB EXPO 2017
Fixing the failure and Unit Testing

Repeat & isolate the problem

Fix faulty components

Simulation Scenario

Repeat

Isolate

Fix

Check

Test
State Synchronization Error

Engine Turning Backwards

Turning backwards

Isolate and Fix
Unit Testing Workflow

1. Separate
2. Reproduce Bug and Fix
3. Confirm
4. Synchronize

MATLAB EXPO 2017
In-model Verification

Isolate Component in Test Harness

Verification Result streamed to Data Inspector

In-model Verification with `verify` keyword

MATLAB EXPO 2017
Check for further design errors

Simulation Scenario

Repeat & isolate the problem

Fix faulty components

... Check everything
Develop a Robust Design with Static Checking

Static Analysis Capabilities

- Simulink Design Verifier
  - Property Proving
  - Design Error Detection
  - Test Generation

- Model Advisor
- Model Metrics
Find Hidden Issues with Design Error Detection

Transition not possible given range of $u$, $h1$, $h2$

$U: [0, 10]$
$h1 = 15$
$h2 = 5$
Fixing as you go Demo

\[ F_{req} \times \text{VehSpd}/3.6 < P_{batt\_discharge\_limit} \land \land \text{AccPed} < 0.9 \]

\[ F_{req} \times \text{VehSpd}/3.6 > P_{batt\_discharge\_limit} \lor \lor \text{AccPed} > 0.9 \]

\[ \text{SOC} \leq \text{SOC\_CSLimit} \]

\[ \text{EVArea, EngArea} = \text{OperatingAreaCalc}(F_{req}, \text{VehSpd}) \]

\[ F_{req} \times \text{VehSpd}/3.6 > P_{batt\_discharge\_limit} \lor \lor \text{AccPed} \geq 0.9 \]

\[ \text{HEV} \]

\[ \text{Simulink Function} \]

\[ \text{EVArea, EngArea} = \text{OperatingAreaCalc}(F_{req}, \text{VehSpd}) \]
Prevent errors by Fixing-as-you go

- **Edit-time checking**
  - Simulink
  - Stateflow
  - Modeling Standards
    - Prohibited blocks violations
    - Block and port name violations

- **Customize rules to corporate standards**

![Invalid port name](image1)

![Missing Default Transition](image2)

![Duplicate Data Store](image3)
Test : Systematically and Completely

Repeat & isolate the problem
Fix faulty components
Check everything

Test Systematically and Test Completely
Test Manager Platform
Systematic authoring, management, execution, and reporting of test cases

- Unites together a broad set of capabilities
- Simulink Test
- R2015a
function customCriteria(test)

% criteria 1: State of charge should not be below 30%
minSOC = min(test.SimOut.get('tmp_raccel_logout').get('BattSignal'));
test.verifyGreaterThan(minSOC, 30, 'SOC should be more than 30%');

% criteria 2: Voltage should not be less than 250 V after starting
minVoltage = min(test.SimOut.get('tmp_raccel_logout').get('BattSignal'));
test.verifyGreaterThan(minVoltage, 250, 'Voltage should be more than');
Top-It-Off Workflow

1. Run Existing Tests
   - Aggregate Coverage

2. Simulink Design Verifier
   - Generate Test Cases

3. Run New Tests
   - Aggregate Coverage
### AGGREGATED COVERAGE RESULTS

**ANALYZED MODEL**
- HEV_MultMode_Optim_R2016a_r1
- MultiModeCtrl_KO_R2016a_r3_end
- Power_Management_v0

<table>
<thead>
<tr>
<th>Analyzed Model</th>
<th>Report</th>
<th>Comp.</th>
<th>D1</th>
<th>C1</th>
<th>Mode</th>
<th>Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEV_MultMode_Optim_R2016a_r1</td>
<td>389</td>
<td>54%</td>
<td>81%</td>
<td>48%</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>MultiModeCtrl_KO_R2016a_r3_end</td>
<td>143</td>
<td>76%</td>
<td>62%</td>
<td>48%</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>Power_Management_v0</td>
<td>49</td>
<td>91%</td>
<td>97%</td>
<td>48%</td>
<td>100%</td>
<td></td>
</tr>
</tbody>
</table>

---

### Report Generated by Test Manager

**Title:** Test
**Author:** David Bolsay
**Date:** 24-May-2016 13:09:13

**Test Environment**
- **Platform:** PCWIN64
- **MATLAB:** R2016a Prerelease

**Properties**
- **Start Time:** 05/23/2016 12:45:45
- **End Time:** 05/23/2016 12:48:36
- **Type:** Baseline Test
- **Test File Location:** C:\work\matlab2016\HEV_Co...
- **Model:** HEV_MultMode_Optim_R...
- **Simulation Mode:** accelerator
- **Test Case Definition:**
- **Baseline File:** C:\work\matlab2016\baseline...
Any continuous integration system that supports Test Anything Protocol (TAP)
Code-to-Model Verification

- Reuse model-based tests
- Equivalence testing
- Code Coverage

Test Manager

Software-In-Loop (SIL)

Code Generator

C

Dynamic

Polyspace

Static
Equivalence Checking and Code Coverage (Software-In-Loop)
Justified

Function: MultiModeCtrl_R0_R2016a_r3_exe2_all_step
Model Object: Rate Limiter3
Metric: Decision (D1) Coverage: 100% ((1-1)/2) decision outcomes

Decisions analyzed:
- clutemonthspd > 250.0F: 0/101 true
- false 101/101
Continuous Test and Verification Framework helps to...

"Reactively" Reproduce the Field Issues in Simulation

Repeat
- Reproduce the failure in simulation

Slice
- Isolate the problematic behavior

Fix
- Fix and Perform Unit Testing

Check
- Check for further design errors

Test
- Test Systematically and Test Completely

"Proactively" Prove that Implementation satisfies Requirements

Check
- Check for design errors early

Fix
- Fix and Perform Unit Testing

Slice
- Isolate the problematic behavior to simplify debugging

Test
- Test Systematically and Test Completely

Prove
- Prove Safety Properties/Requirements
Model Based Design helps to…

Productivity + Quality

Continuous Test & Verification

Check
Fix
Slice
Test
Prove

Repeat
Slice
Fix
Check
Test
MathWorks Training Offerings

Verification and Validation of Simulink Models

ADVANCED

This one-day course describes techniques for testing Simulink model behavior against system requirements. Topics include:

- Identifying the role of verification and validation in Model-Based Design
- Creating test cases for Simulink models
- Analyzing simulation results to verify model behavior
- Automating testing activities and managing results
- Formally verifying model behavior
- Automatically generating artifacts to communicate results

Prerequisites: MATLAB Fundamentals and Simulink for System and Algorithm Modeling. This course is intended for intermediate or advanced Simulink users.

MATLAB EXPO 2017  http://www.mathworks.com/services/training/
Speaker Details

Manohar Reddy

Email: Manohar.Reddy@mathworks.in

Contact MathWorks India

Call: 080-6632-6000

Email: info@mathworks.in