Continuous Model & Code

System & Component

Dynamic testing & Static analysis

Test & Verification

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

 subsystems: Battery, Engine, Energy Management

Controller Vehicle

Simulink Plant Model

Production C-Code

Failure

Complete Integration & Test

System-Level Integration & Test

Subsystem Integration & Test
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
Modeling the Test

Drive cycle

Scenario

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%
Model Slicing: Isolate the problematic behavior
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

MATLAB EXPO 2017
Model Slicing
Complete Model Slicer Workflow

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

Simulation Scenario → Controller → Plant → Model Slicer → Area of Interest

Highlight of Relevant Parts

Stateflow Highlighting R2016b

Identify Interest

Original Design Model

Create Model

Highlighted Model

Iterate

Slice Model

Updated Slice Model

Incorporate Changes

MBD Work

MATLAB EXPO 2017
State Synchronization Error

Engine Turning Backwards

Isolate and Fix

MATLAB EXPO 2017
Unit Testing Workflow

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

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 $u$, $h_1$, $h_2$

$U: [0,10]$
$h_1 = 15$
$h_2 = 5$
Fixing as you go Demo

1. \(\frac{F_{req} \cdot VehSpd}{3.6} < P_{batt\_discharge\_limit} && AccPed < 0.9\)

2. \(SOC <= SOC_{CSLimit}\)

3. \(\frac{F_{req} \cdot VehSpd}{3.6} > P_{batt\_discharge\_limit} || AccPed > 0.9\)

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

[HEV]

Simulink Function
\[\text{Simulink Function}\]
\[\text{[EVArea, EngArea]} = OperatingAreaCalc(F_{req}, VehSpd)\]
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**
Test : Systematically and Completely

Repeat & isolate the problem

... Fix faulty components

... Check everything

... Test **Systematically and** Test **Completely**

MATLAB EXPO 2017
Test Manager Platform

Systematic authoring, management, execution, and reporting of test cases

- Unites together a broad set of capabilities
- Simulink Test
- R2015a
MATLAB Script Criteria

function customCriteria(test)

% criteria 1: State of charge should not be below 30%
minSOC = min(test.SimOut('tmp_racel_logsoot').get('BattS

% criteria 2: Voltage should not be less than 250 V after starting
minVoltage = min(test.SimOut('tmp_racel_logsoot').get('Bat"
Top-It-Off Workflow

1. Run Existing Tests
   - Aggregate Coverage

2. Simulink Design Verifier
   - Generate Test Cases

3. Run New Tests
   - Aggregate Coverage

R2017a
Integration with Test Automation Servers

Any continuous integration system that supports Test Anything Protocol (TAP)
Code-to-Model Verification

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

Code Generator

Test Manager

Software-In-Loop (SIL)

Dynamic

Static

Polyspace
Equivalence Checking and Code Coverage (Software-In-Loop)
Justification for Code Coverage

10.28. Decision cltn0natspd > 250.0 (line 2769)

**Justify or Exclude**
- **Function:** Cltn0natspd
- **Model Object:** Rate Limiter
- **Uncovered Links:** ∗ ∗
- **Metric:** Coverage
- **Decision (D1):** 50% (1/2) decision outcomes

Decisions analyzed:
- cltn0natspd > 250.0 (50%)
- false: 101/101
- true: 0/101

10.28. Decision cltn0natspd > 250.0 (line 2769)

**Justified**
- **Function:** Cltn0natspd
- **Model Object:** Rate Limiter
- **Metric:** Coverage
- **Decision (D1):** 100% ((1+1)/2) decision outcomes

Decisions analyzed:
- cltn0natspd > 250.0 (50%)
- false: 101/101
- true: 0/101
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

MATLAB EXPO 2017
Model Based Design helps to…

Productivity

Quality

Continuous Test & Verification
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.

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