Model-Based Engineering Platform to Manage Complexity and Scale

Ramamurthy Mani
Engineering Director, Simulink Semantics
Our theme today: Evolution
The Three Evolutionary Forces at Play

1. Simulation Scale
2. Design Complexity
3. Collaborative Engineering
Evolving for **Simulation Scale**

**BRAIN SIZE AND NEURON COUNT**
Cerebral cortex mass and neuron count for various mammals.

<table>
<thead>
<tr>
<th></th>
<th>Copybara</th>
<th>Rhesus Macaque</th>
<th>Western Gorilla</th>
<th>Human</th>
<th>African Bush Elephant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass (g)</td>
<td>48.2</td>
<td>69.8</td>
<td>377</td>
<td>1232</td>
<td>28.48</td>
</tr>
<tr>
<td>Neurons</td>
<td>0.3 billion</td>
<td>1.71 billion</td>
<td>9.1 billion</td>
<td>16.3 billion</td>
<td>5.59 billion</td>
</tr>
</tbody>
</table>

Trend: Demand for scaled up simulation capabilities

Full Vehicle Simulation
Strategy: Continuously evolve Simulink to be a best in class Simulation Integration Platform

Windows

Linux

Simscape

3rd party tool

3rd party tool

MATLAB

Deep Learning NN

S-Function

Hand C/C++ Code

fmu
The primary challenges for simulation scale

Integration
Performance
Operationalization
Integration of algorithms with multiple simulation interfaces is key.

Integration Platform for Models, Code, Performance, Other tools, Operationalization.
For Models, core modularity principles underpin integration

Data Encapsulation

Interface Management

SLDD

[\text{Ts}] = 0.1
NCyl = 4

SLDD

\text{Common types, constants, …}

Copyright 2019 The MathWorks, Inc.
You can easily bring C/C++ code into Simulink

```c
void function_name() {
   ...
   ...

   ...
}
```

---

Basic

- **C Caller**
  - u
  - getLimits
  - y

- **C Function**
  - inData
  - sum
  - mean

Advanced

- **S-Function Builder**
  - system

- **S-Function**
  - R2018b
  - R2020a
You can use MATLAB algorithms like the Deep Learning Toolbox in Simulink

```matlab
obj.DLModel = coder.loadDeepLearningNetwork('mydnn.mat', 'network')
```

Deep Learning Toolbox

Tensorflow-Keras Importer

Keras TensorFlow
Simulink has simulation interfaces to 190 connection partner products and services primarily through the S-Function interface.
Using FMUs inside Simulink is easy and expressive
System-level simulations are computationally expensive

Integrating models and components

Simulation Performance

Maximize single thread performance

Exploit parallelism

Enable massive simulation runs

Operationalization
Maximizing performance by discovering speed-up opportunities: Performance Advisor

- Consolidated advice on performance
- Gives advice that works!
- Helps discover performance focused capabilities
Invest in multiple parallelization techniques for boosting performance

Model block, S-function, FMU import

R2018a

Dataflow SIMD

ForEach Subsystem Parallelization

MATLAB Function GPU acceleration

Compute Clusters
Design envelope studies require a large number of simulations

Full vehicle model
100 drive cycles \times 10 \text{ vehicle loadings} \times 10 \text{ weather conditions}

Driving cycle

Optimize gear ratios
Simulink enables massive simulation workflows

**Setup**

**Simulate**

Simulation Manager

**Analyze**

Simulation Manager

parsim

batchsim
Extend simulations to Operational phases of the system
Simulink Compiler enables deployment of simulations
Trend: Demand for simulating complex scenarios with multiple actors is increasing

Scenario Simulations for Autonomy
Strategy: Create a platform for system-of-systems (scenario) simulations

- Virtual Testbed
  - Scenarios
  - Sensor Models
  - Application under Test
  - Metrics
  - Performance Evaluator
  - Test Manager

- Cuboid Driving Simulation
- Unreal Engine Driving Simulation
- RoadRunner
Simulink platform is evolving to meet the demands of scaled up simulations

Full Vehicle Simulation

Scenario Simulations for Autonomy

Integrating models and components

Simulation Performance

Operationalization

Scenario Simulation
Evolving for **Design Complexity**

[Image: https://en.wikipedia.org/wiki/Tiktaalik]
Trend: Some rumblings in the force

MAB Breakout session 2012 on System Architecture

“Not sure you get it…”

Wonder what’s for lunch?
Why the discontent?

Customer quote:
“We have tried to build the architecture model in SysML and connect it to the design in Simulink … … does not work without rework both in the architecture and design worlds whenever a change is needed. It is broken and we need a more integrated approach”
Are you happy with your current tool choices for Modeling System Architecture?
More specifically, what are the pains? “We do not like our current System Architecture solution because they are:”

Newton MAB Survey 2019

- Not Executable
- Not synchronized with designs
- Hard to use
- Not analyzeable
Strategy: Build an MBSE Ecosystem that fits with MBD

- Be Intuitive
- Facilitate Analysis
- Tackle Complexity
- Enable Implementation

**System Composer**

**MATLAB**

**Simulink**

Requirements Coverage Reporting and Impact Analysis

**Simulink Requirements**
“Sketch” system interfaces and elaborate incrementally
Extend elements with your own custom metadata using Profiles & Stereotypes
Analyze system characteristics and quantitatively evaluate choices using MATLAB.

- **Endurance**: 4.0997877
- **Mass**: 85
- **PowerDraw**: 40
Trace to system requirements
Refine requirements alongside the architecture
Link design models to components and ensure consistent interfaces
Simplify the complex with Filters and autogenerated Views
Simplify the complex with Filters and autogenerated Views

Stereotype isa ElectricalComponent
And we are only getting started. Coming soon:

- Behavior modeling using Sequence Diagrams
- Architecture Allocations though Analysis (e.g. Logical to Physical)
- Software Architecture Modeling
  - Link to AUTOSAR (R2019b)
  - Other middlewares such as DDS
- And much more!
Evolving for **Collaborative Engineering**

https://en.wikipedia.org/wiki/Symbiogenesis
Trend: An increased demand for Agile team-based workflows

- Shared team environment
- Collaboration
- Continuous Integration & Test

Simulate

Jenkins
Strategy: Continued investments to facilitate Continuous Integration as a critical lynch-pin in Agile workflows
Can I do CI today in Simulink?
Yes, let's consider an example from **R2019b**
Lane Following Assist Example

Simulink Check Checks
SIL Code Generation
SIL Testing Simulink Test
How Does It All Fit Together?

Simulink Check Checks
Generate Controller Code
Execute Tests in Simulink Test

Developers & Test Authors

Development
Review
Develop
Test
Merge
Submit
1. Trigger
1. Trigger

Continuous Integration
1. Trigger

Simulink Check

Code Generation

## Caching model source code

### Writing header file rtGetNaN.h

### Writing source file rtGetNaN.cpp

### Writing header file rt_defines.h

### Writing header file rt_nonfinite.h

### Writing source file rt_nonfinite.cpp

Running LaneFollowingModelAdvisorChecks
Done LaneFollowingModelAdvisorCheck
2. Detect

Failure Summary:

<table>
<thead>
<tr>
<th>Name</th>
<th>Failed</th>
<th>Incomplete</th>
<th>Reason(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaneFollowingTestScenarios &gt; Scenarios/LFACC_Curve_CutInOut_TooClose</td>
<td></td>
<td>X</td>
<td>Failed by verification.</td>
</tr>
</tbody>
</table>

ERROR: MATLAB error Exit Status: 0x00000001
Build step 'Run MATLAB Tests' changed build result to FAILURE
Finished: FAILURE
3. Reproduce

- Continuous Integration
- Build
- Develop
- Developers & Test Authors
- Reproduce Locally
- Test
- Verify
- Trigger

![Diagram showing the reproduction process in continuous integration pipeline]

- Developers & Test Authors
  - Reproduce Locally
  - Build
  - Continuous Integration
  - Trigger
  - Verify

- Diagram showing test management and integration process:
  - Test Manager
  - Data Inspector
  - Format
  - Test Browser
  - Visualize
  - Test Assessments/Global: verify/duration/time_gap < 1.0, sec < 3
4. Fix Locally

Continuous Integration

Reproduce Locally

Development

Build

Test

Developers & Test Authors

Globl Assessments

% Ensure that the time gap between the ego vehicle and lead vehicle does not dip below 1.5s for more than 2s at a time.
verify(duration(time_gap < 1.5, sec) < 2);

% Verify that no collision was detected
verify(!collision);

% Verify that the absolute value of lateral deviation from the lane centerline does not exceed 0.2m
% for more than 5s at a time.
verify(duration(abs(lateral_deviation) > 0.2, sec) < 5);

Globl Assessments

% Ensure that the time gap between the ego vehicle and lead vehicle does not dip below 0.8s for more than 5s at a time.
verify(duration(time_gap < 0.8, sec) < 6);

% Verify that no collision was detected
verify(!collision);

% Verify that the absolute value of lateral deviation from the lane centerline does not exceed 0.2m
% for more than 5s at a time.
verify(duration(abs(lateral_deviation) > 0.2, sec) < 5);
5. Test Locally

- Continuous Integration
- Build
- Test
- Reproduce Locally
- Development
- Developers & Test Authors
- Trigger
- Verify
- Test
- Locally
- Reproduce
- Development
- Test
- Continuous Integration

Image: Diagram illustrating the Test Locally process with steps such as trigger, verify, build, test, reproduce locally, and development.
6. Merge

Continuous Integration

Build

Test

Locally

Merges

Development

Test

Verify

Developers & Test Authors

Trigger

Developers & Test Authors

Verify

Continuous Integration

Build

Test

Locally

Merges

Development

Test

Verify

Developers & Test Authors

Trigger

Continuous Integration

Build

Test

Locally

Merges

Development

Test

Verify

Developers & Test Authors

Trigger
6. Review

Developers & Test Authors

Continuous Integration

Test

Reproduce Locally

Develop

Development

Mere

Test

Verify

Trigger

GlobalAssessments

labelString : GlobalAssessments

% Ensure that the time gap between the ego vehicle and lead vehicle does not dip below
% 1.5s for more than 2s at a time.
verify(duration(time_gap < 0.8, sec) < 2);

% Verify that no collision was detected
verify(~collision);

GlobalAssessments

labelString : GlobalAssessments

% Ensure that the time gap between the ego vehicle and lead vehicle does not dip below
% 1.5s for more than 5s at a time.
verify(duration(time_gap < 0.8, sec) < 5);

% Verify that no collision was detected
verify(~collision);
7. Commit

- Continuous Integration
- Build
- Test
- Reproduce Locally
- Develop
- Review
- Merge
- Development
- Test
- Submit
- Version Control

---

**Update Safe Distance assessment criteria to pass if the time gap does not dip below 0.8s for more than 5 seconds at a time.**

- **Submit**
- **Cancel**
8. Verify, Build, Test

Finished: SUCCESS
Continuous Integration Success is within your reach

1. Tooling
   - Jenkins Plugin
   - MATLAB Unit
   - Simulink Test

2. Documentation
   - Technical Article
   - Documentation Hub
   - Solutions Page

3. Future
   - Pipeline
   - Server Workflows
   - Test Results Online
   - Dashboards
Lets go back to the broad forces that shape our platform evolution

1. Simulation Scale

YOU!

3. Collaborative Engineering
Please contact us with questions

mani@mathworks.com