Model-Based Engineering Platform to Manage Complexity and Scale

Ramamurthy Mani
Engineering Director, Simulink Semantics
The MathWorks Inc.

June 30, 2020 | Europe

MathWorks
AUTOMOTIVE CONFERENCE 2020
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>Cerebral mass</td>
<td>non-primate</td>
<td>primate</td>
<td>primate</td>
<td>primate</td>
<td>non-primate</td>
</tr>
<tr>
<td>(g)</td>
<td>48.2</td>
<td>69.8</td>
<td>377</td>
<td>1232</td>
<td>2848</td>
</tr>
<tr>
<td>Neuron count</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

3rd party tool

Simscape

Windows

Linux

3rd party tool

MATLAB

Deep Learning NN

S-Function

Hand C/C++ Code

Hand C/C++ Code
The primary challenges for simulation scale

Integration

Performance

Operationalization
Integration of algorithms with multiple simulation interfaces is key.
For Models, core modularity principles underpin integration

Data Encapsulation

Interface Management

SLDD

Common types, constants, …

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

SLDD

Model reference
Subsystem reference
Libraries

Componentization

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

```
void function_name() {
    ... ...
    ... ...
}
```

Basic

Advanced

- **C Caller**
- **C Function**
- **S-Function Builder**
- **S-Function**
You can use MATLAB algorithms like the Deep Learning Toolbox in Simulink.

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

Host 1

Host 3

ForEach Subsystem Parallelization
MATLAB Function GPU acceleration
Compute Clusters

Dataflow SIMD

R2018a

R2018a
Design envelope studies require a large number of simulations

- Full vehicle model
  - 100 drive cycles × 10 vehicle loadings × 10 weather conditions
  - 10,000 simulations

- Optimize gear ratios
  - 100,000 simulations

- Driving cycle

Copyright 2019 The MathWorks, Inc.
Simulink enables massive simulation workflows

**Setup**

**Simulate**

Simulation Manager

**Analyze**

Simulation Manager

parsim

batchsim
Extend simulations to Operational phases of the system

Integrating models and components

Simulation Performance

Operationalization

Enable simulation deployment
Simulink Compiler enables deployment of simulations

Simulink Compiler

R2020a

Integrate as Enterprise Application

Web App
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
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**

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?
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

With Simulink Requirements
Link design models to components and ensure consistent interfaces
Simplify the complex with Filters and autogenerated Views

VIEW BROWSER
- Electrical View
- Basic Elements View
- Motors View
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 through 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
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?

Development

Review

Develop

Test

Merge

Simulink Check Checks

Generate Controller Code

Execute Tests in Simulink Test

Developers & Test Authors
1. Trigger

Continuous Integration

Running LaneFollowingModelAdvisorChecks
. Done LaneFollowingModelAdvisorCheck

Simulink Check
1. Trigger

Continuous Integration

Running LaneFollowingModelAdvisorChecks.
Done LaneFollowingModelAdvisorCheck

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
## Simulink Test

### 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_CutInOutTooClose</td>
<td></td>
<td></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

Test

Reproduce Locally

Development

Developers & Test Authors

Trigger

Verify
4. Fix Locally

Developers & Test Authors

- Continuous Integration
- Development
- Build
- Test
- Reproduce Locally

Trigger

- Verify

Global Assessments

% Ensure that the time gap between the ego vehicle and lead vehicle does not drop 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);

% Ensure 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);

% Ensure that the time gap between the ego vehicle and lead vehicle does not drop 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

Developers & Test Authors

Build

Test

Repeatable

Locally

Development

Continuous Integration

Verify

Trigger

Test

Reproduce Locally

Verify Test Scenarios

Continuous Integration

Trigger
6. Merge

[Diagram showing the process of merging with steps like Trigger, Verify, Build, Merge, Test, Development, and Continuous Integration. The diagram illustrates the flow of developers and test authors, with arrows indicating the sequence of tasks.]
6. Review

Developers & Test Authors

Continuous Integration

Build

Test

Verify

Develop

Trigger

Reproduce Locally

Test

Merge

Development

Verify

Trigger

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);

LaneFollowingTestBenchExample/Collision Detection/Test Assessments

LaneFollowingTestBenchExample/Collision Detection/Test Assessments

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

Developers & Test Authors

Verify

Reproduce Locally

Test

Build

Merge

Review

Develop

Test

Verify

Commit

SOURCE CONTROL

Enter a Comment

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

**Tooling**

1. Jenkins Plugin
2. MATLAB Unit
3. Simulink Test

**Documentation**

1. Technical Article
2. Documentation Hub
3. Solutions Page

**Future**

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

1. Simulation Scale

YOU!

3. Collaborative Engineering
Q&A

Please contact us with questions

mani@mathworks.com