

Simulink for Virtual Vehicle Development



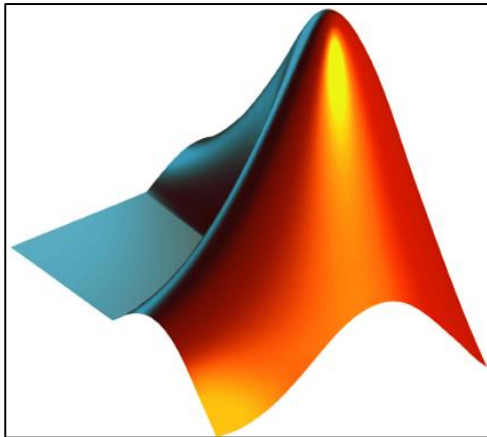
Chris Fillyaw
Application Engineering Manager
cfillyaw@mathworks.com



Mike Sasena, PhD
Automotive Product Manager
msasena@mathworks.com

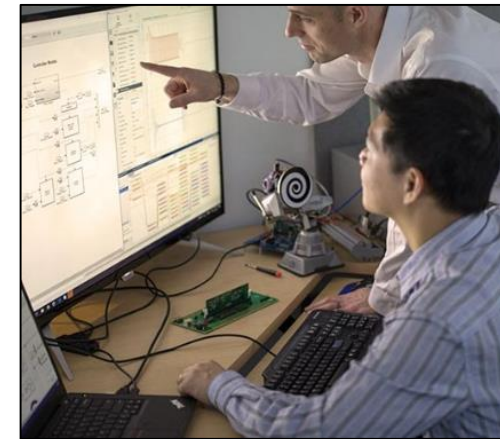
Key takeaways

MathWorks provides a **powerful platform** for building your **Virtual Vehicle**



Out-of-the-box capability

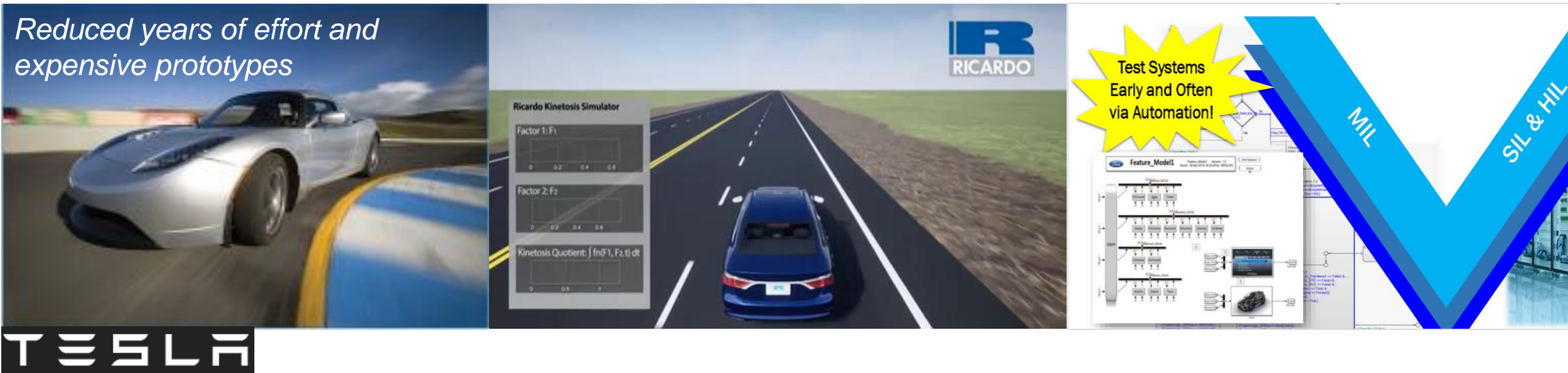
Our platform is very **flexible**, and we can help you **customize** it for your needs



Custom virtual vehicle solution



Virtual vehicle: functional simulation of full vehicle behaviors



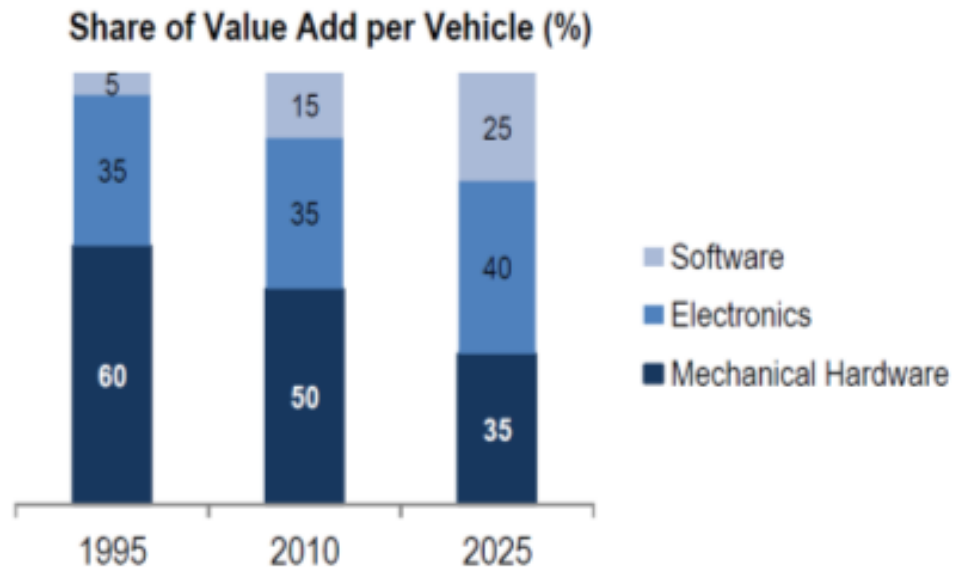
Tesla: vehicle design tradeoff

Ricardo: simulating passenger comfort

Ford: software validation

Reduce physical testing needed before design validation

Embedded software is essential for many virtual vehicle applications



Source: BMW

Virtual vehicle applications such as attribute development, software validation, calibration **require simulation of embedded software.**

- Application software behavior fully represented
- Interfaces consistent with software component definitions
- Basic software components included as need for the application

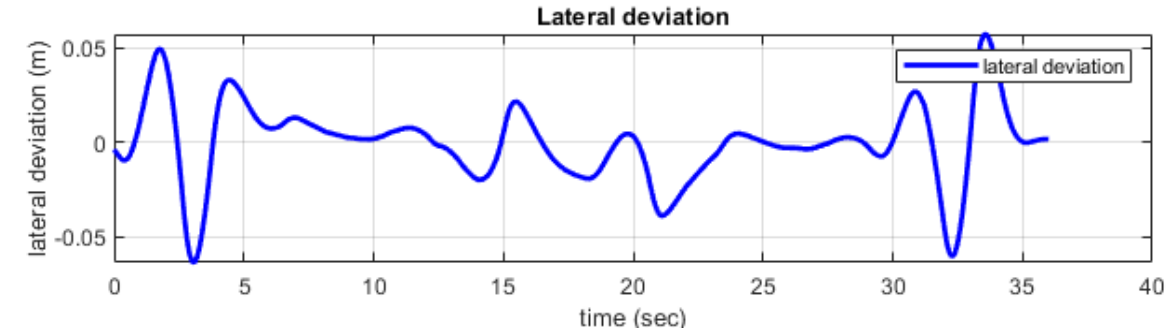
Example: Validating lane following software functional safety requirement (FSR)



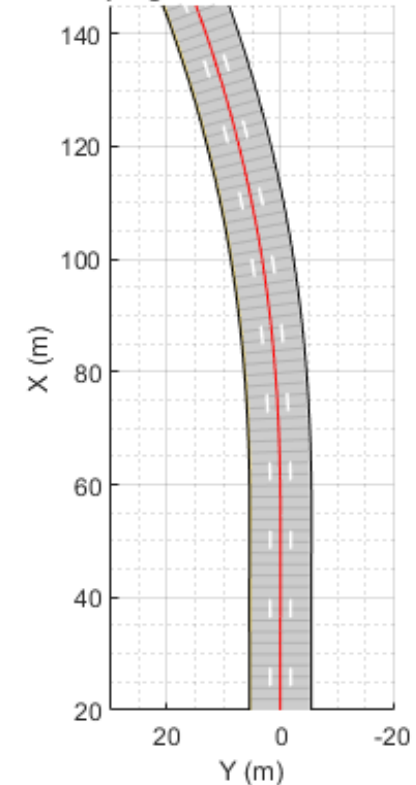
FSR: The lane following system lateral error shall be less than 1 meter

Questions to consider:

- System performance under normal conditions?
- Impact of environment conditions?
- Impact of a component failure?
- Required processor throughput?



Lane keeping assist at curvature change

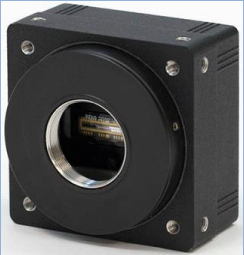


System level interactions need to be considered



FSR: The lane following system lateral error shall be less than 1 meter

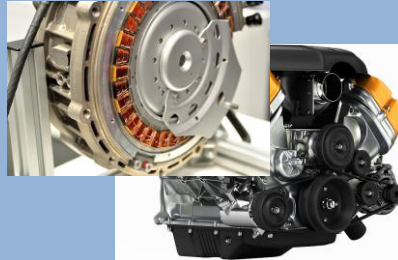
Sensors



Controllers



Powertrain



Environment



Driver



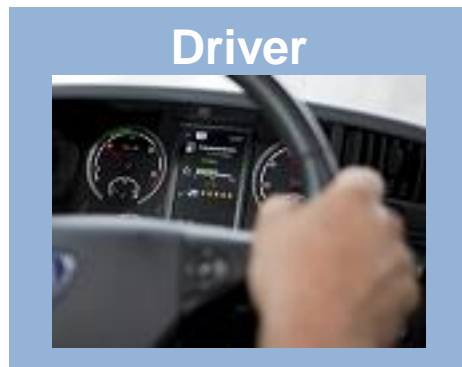
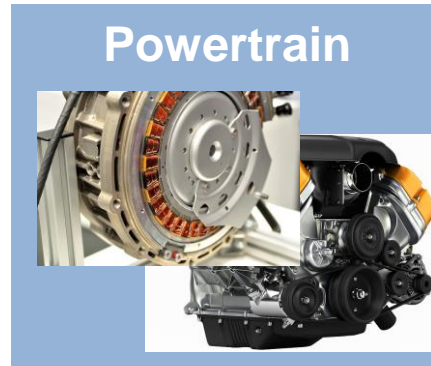
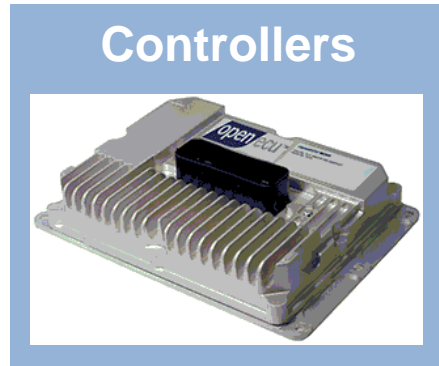
Vehicle



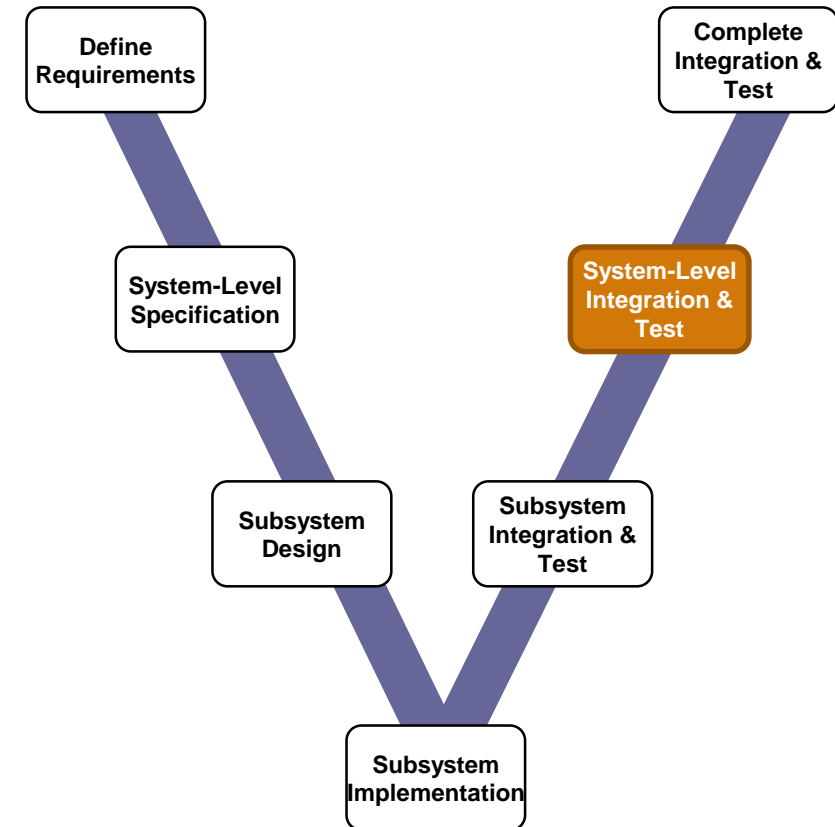
System level testing typically occurs with hardware integration



FSR: The lane following system lateral error shall be less than 1 meter



Discovering problems during system-level integration is expensive



Validate software against function safety requirements early

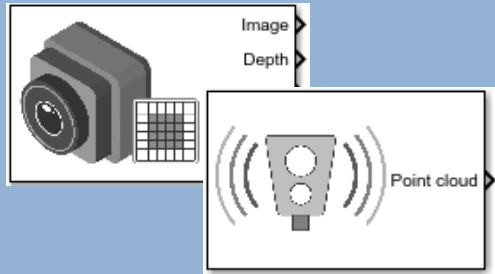


FSR: The lane following system lateral error shall be less than 1 meter

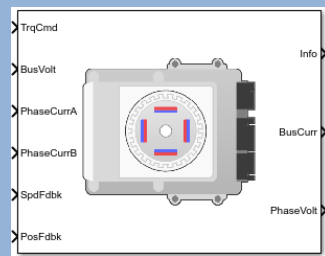
*Use simulation to do system-level integration testing **early***

Virtual vehicle

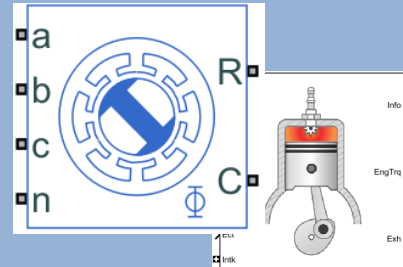
Sensors



Controllers



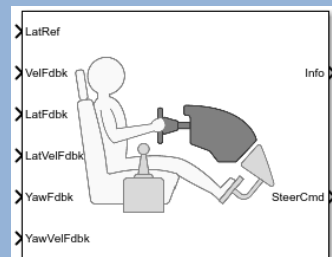
Powertrain



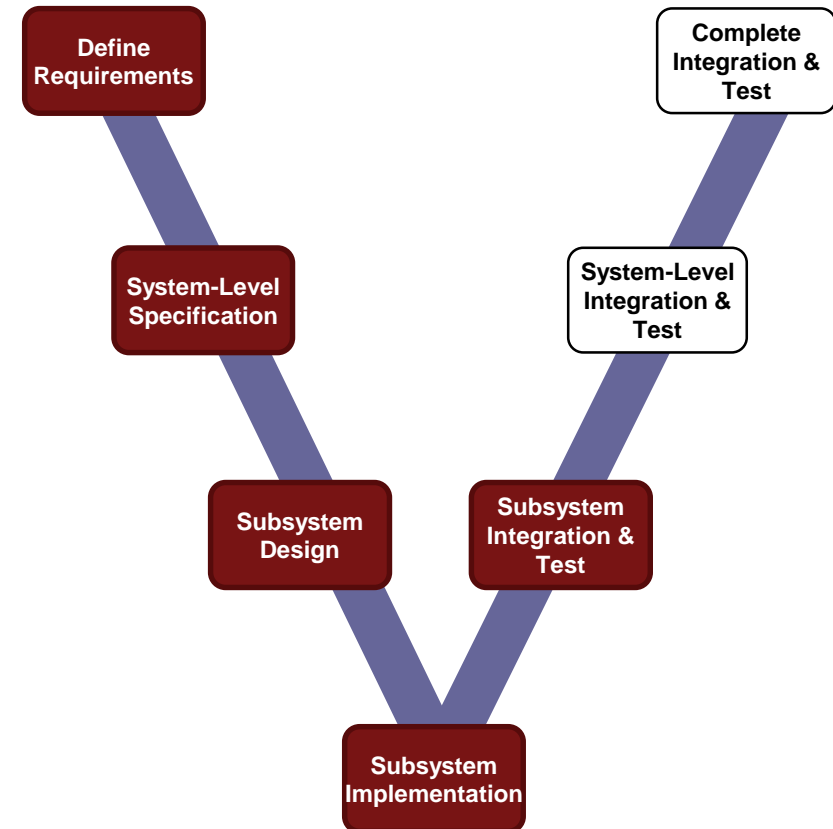
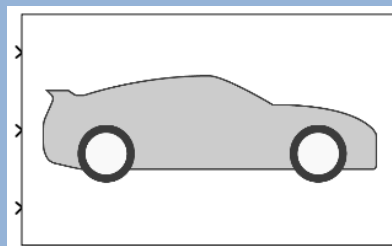
Environment



Driver



Vehicle



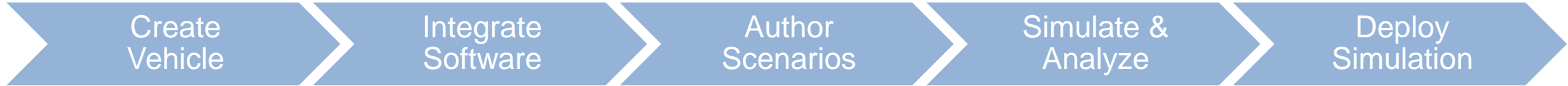
Agenda

- Common challenges
- MathWorks solutions
- Case study

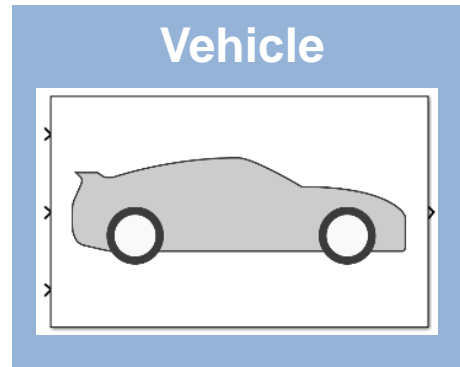
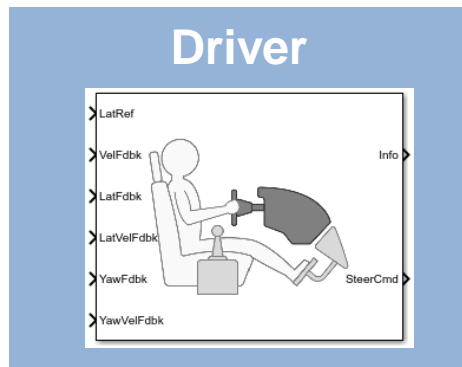
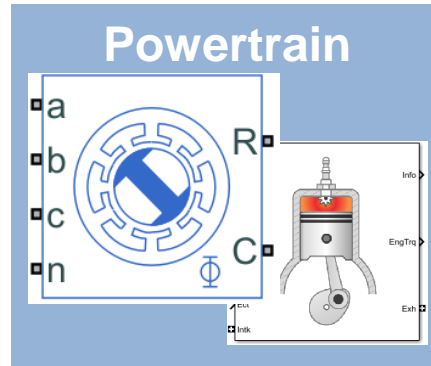
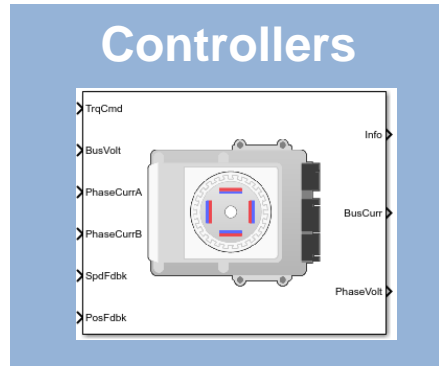
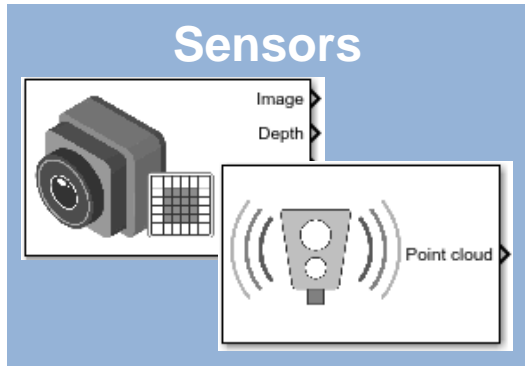
Agenda

- **Common challenges**
- MathWorks solutions
- Case study

Challenges to early system-level testing



Virtual vehicle



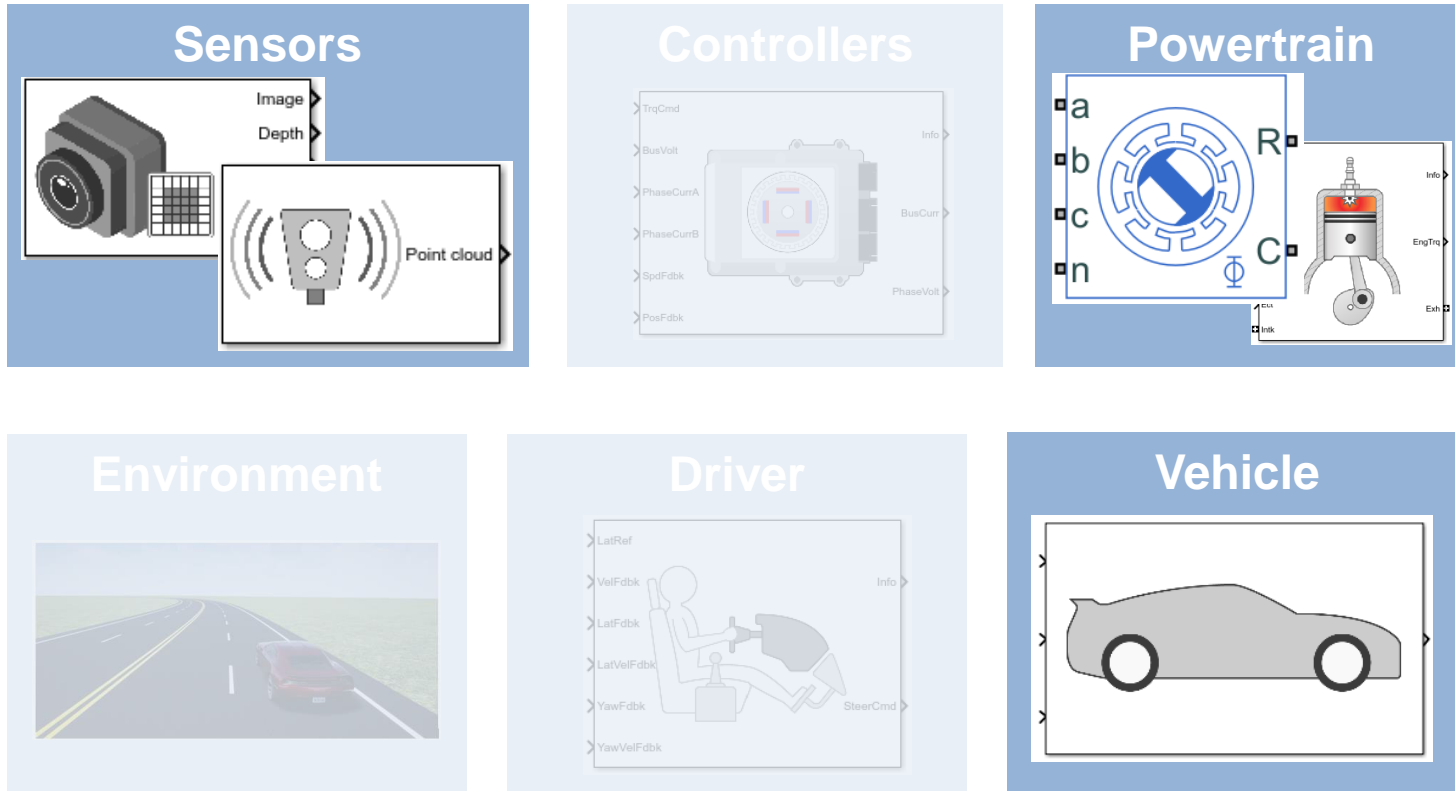
Using a virtual vehicle for systems integration testing early in development can **save time / money**

What are the **challenges** to building one?

Challenges to early system-level testing

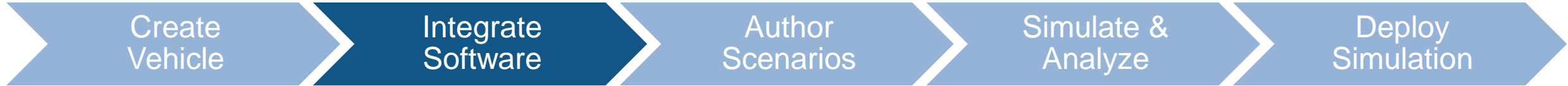


Virtual vehicle

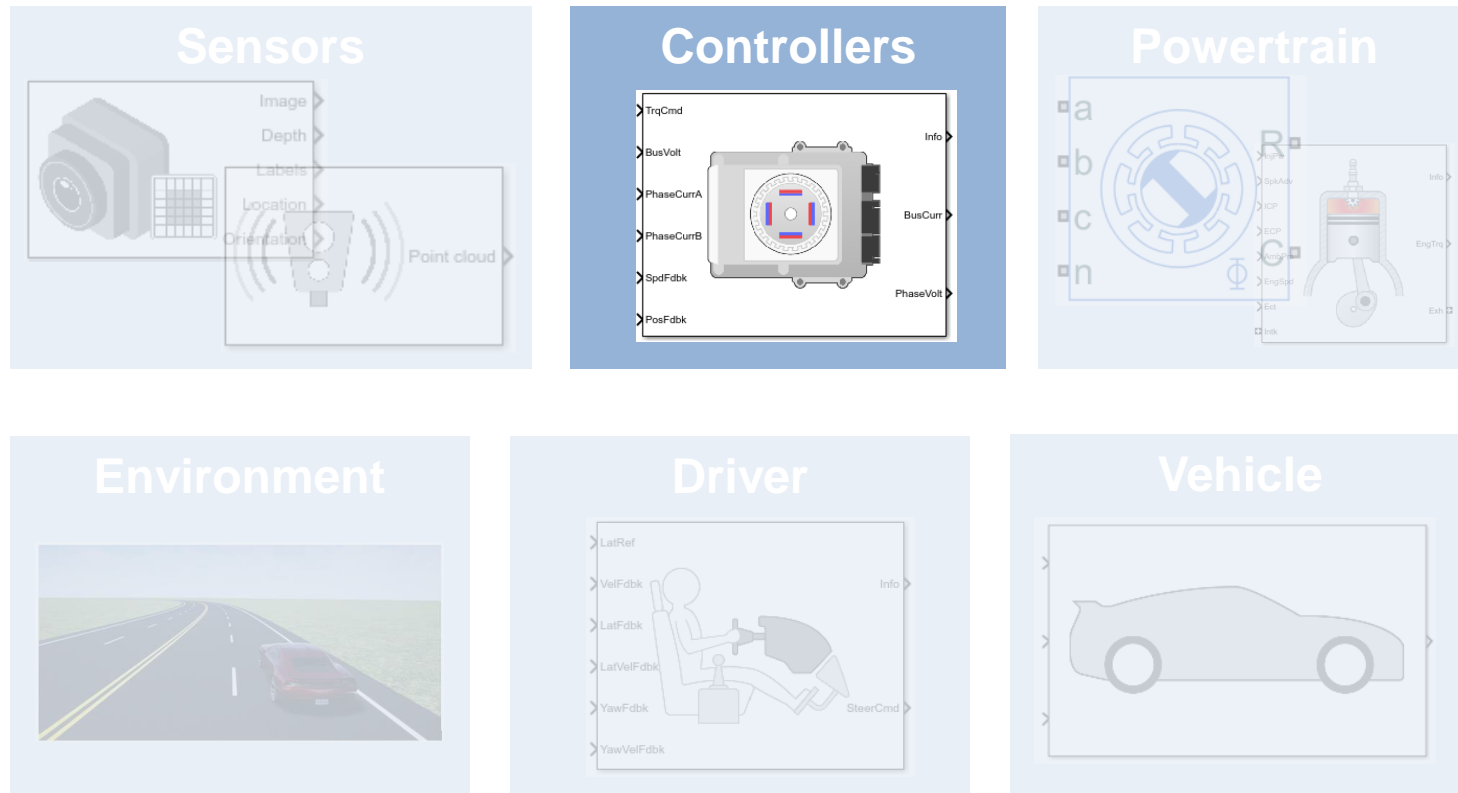


- Availability of appropriate vehicle level model
- Access to plant and sensor models with “right” level of fidelity
- Model calibration

Challenges to early system-level testing



Virtual vehicle

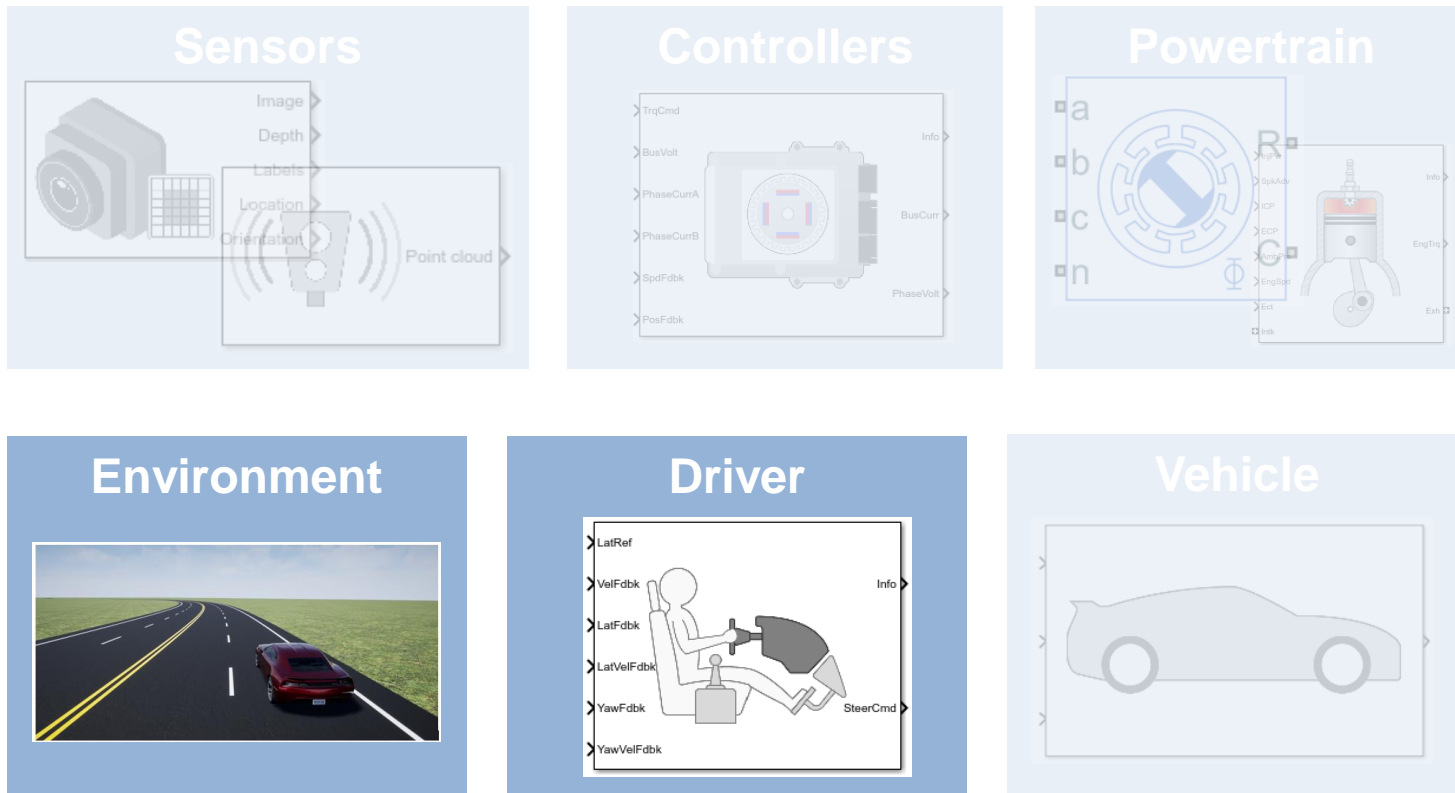


- Standardizing interfaces and data management
- Access to software components across different teams
- Assembly of software components from multiple sources

Challenges to early system-level testing

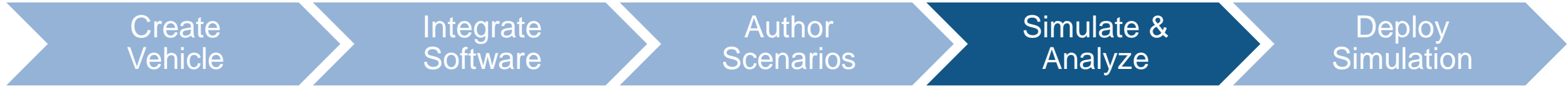


Virtual vehicle

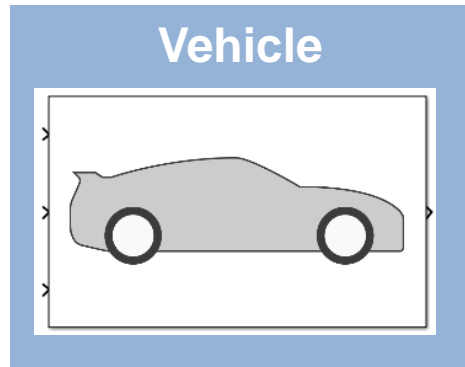
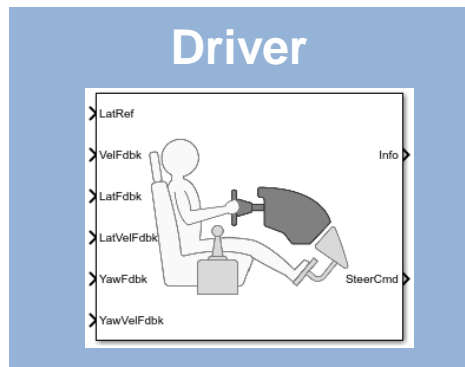
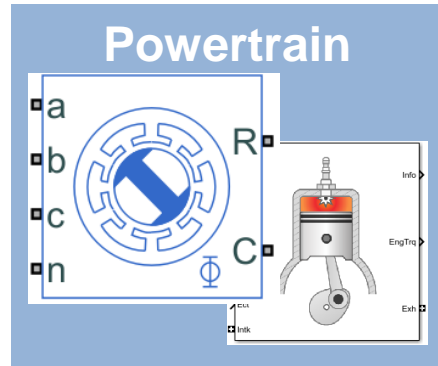
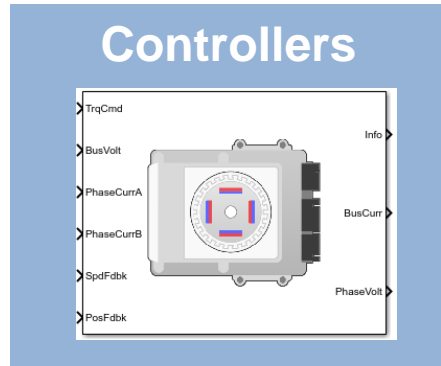
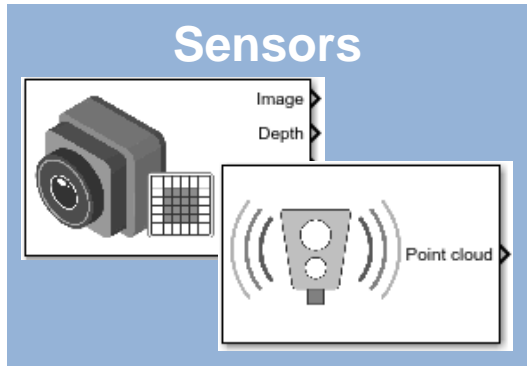


- Creation of virtual 3D environment
- Definition of scenarios to test
- Linking test cases to requirements

Challenges to early system-level testing



Virtual vehicle

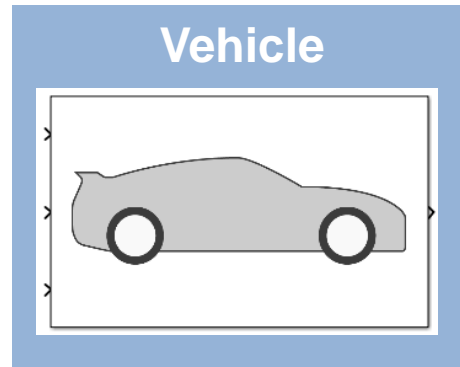
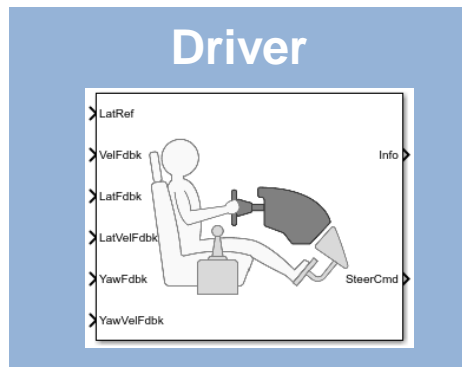
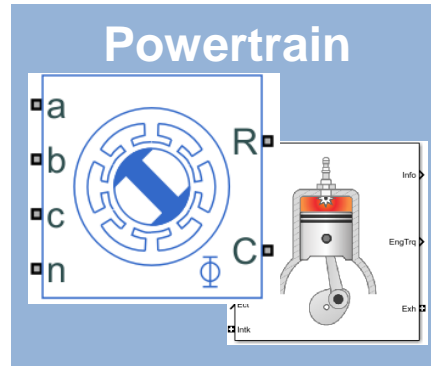
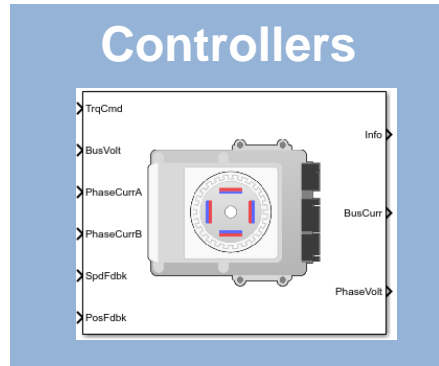
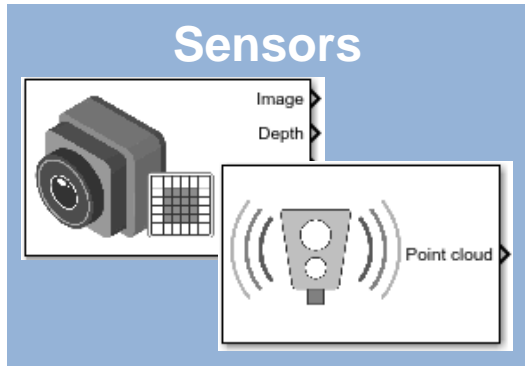


- Post-processing and visualizing results
- Automatically generating reports
- Running large numbers of simulations efficiently

Challenges to early system-level testing



Virtual vehicle

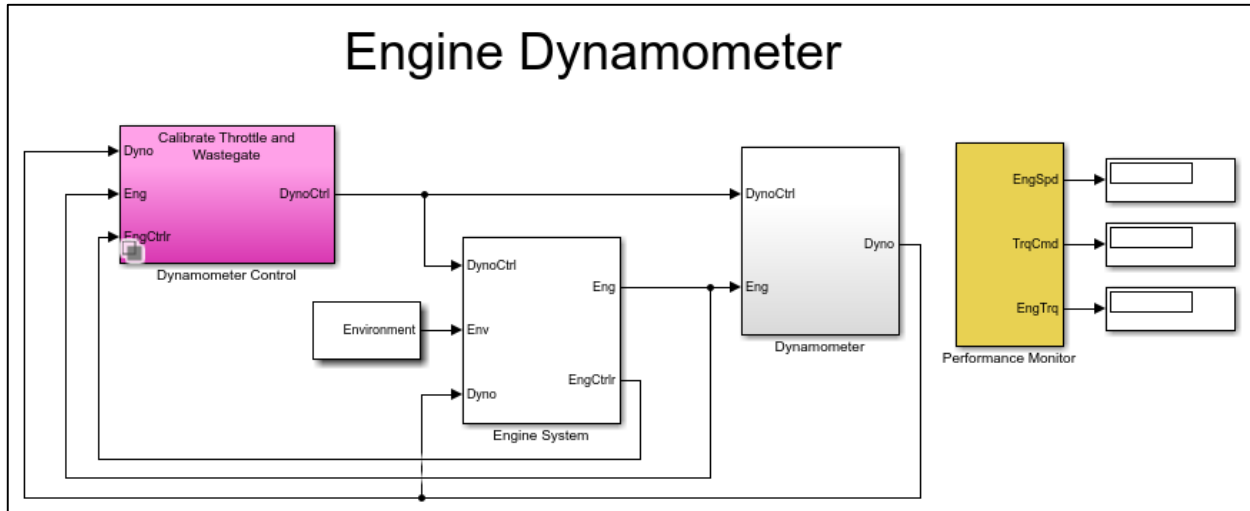


- Sharing models across the organization
- Deploying models to users who aren't tool experts
- Deploying models for SIL, HIL, etc.

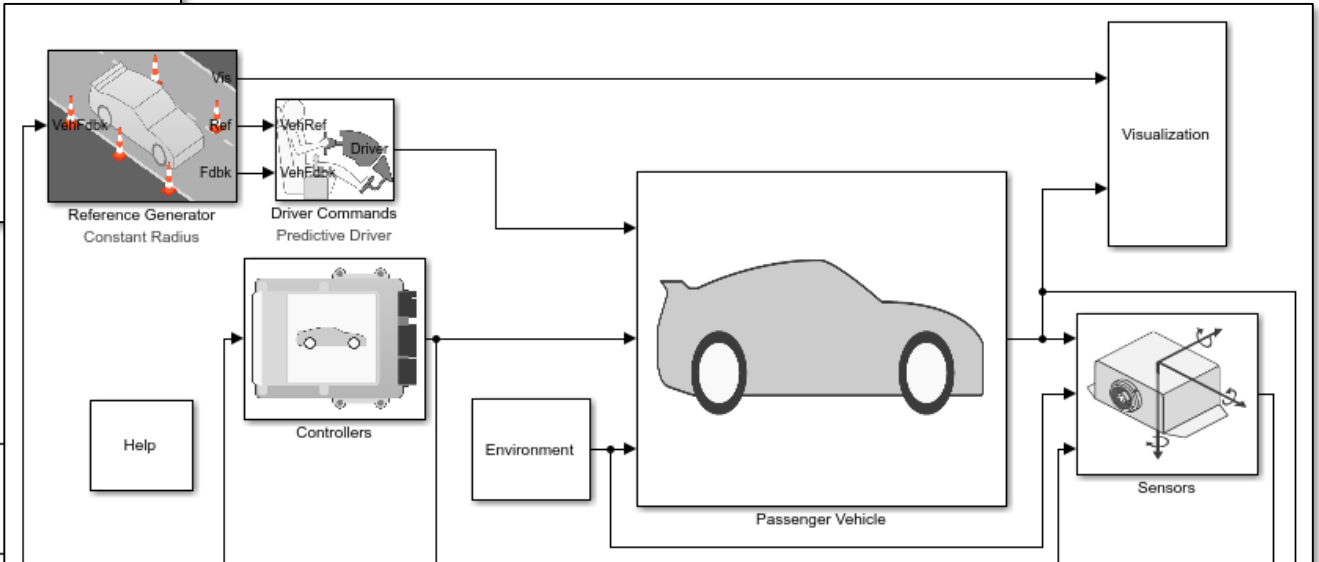
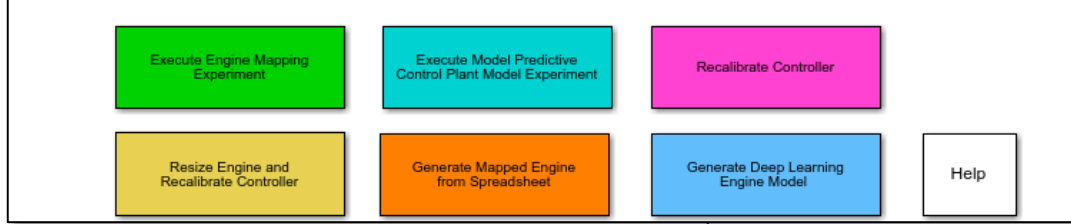
Agenda

- Common challenges
- **MathWorks solutions**
- Case study

MathWorks Virtual Vehicle: reference applications



- Start with in-house vehicle models
 - We can help you customize it and apply best practices for Model-Based Design
- Start with our reference applications
 - Detailed system and vehicle level models for powertrain, vehicle dynamics, ADAS and other applications



Learn more:

[Powertrain Blockset](#)

[Vehicle Dynamics Blockset](#)

[Automated Driving Toolbox](#)

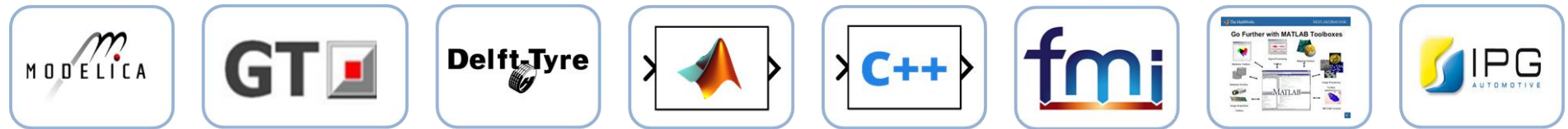
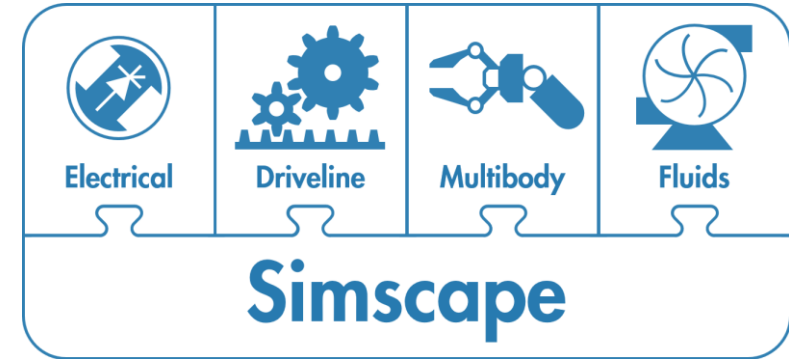
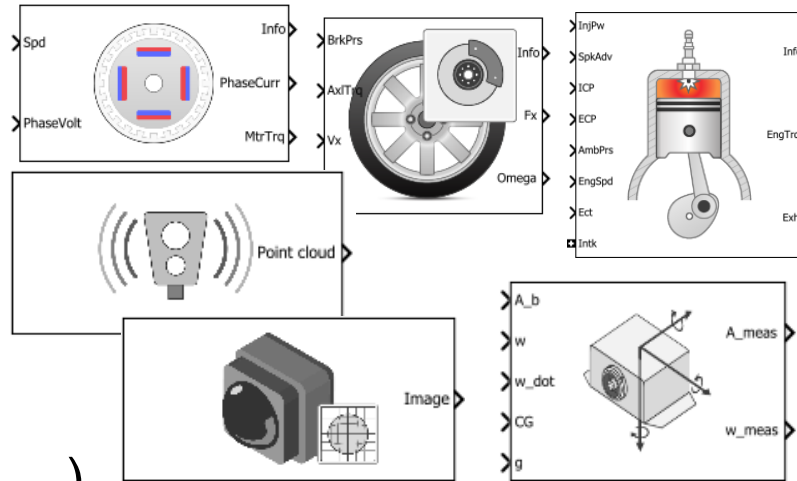
2
BrakeTorque

MathWorks Virtual Vehicle: model customization

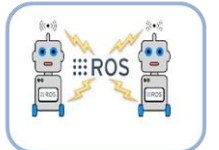
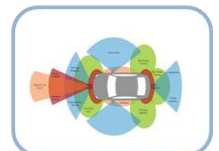
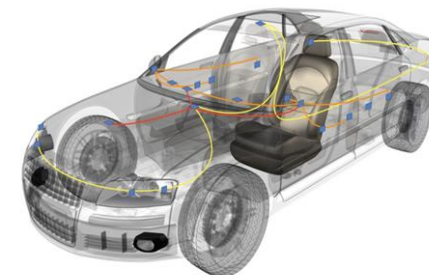
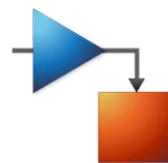


Add detail where needed using:

- In-house Simulink models
- Simulink and Simscape libraries
- 3rd party tools (S-function, FMU, ...)



Simulink

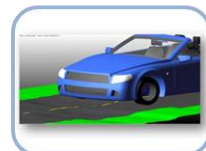


Learn more:

[Simscape](#)

[Multi-core cosim](#)

[Integrate with existing sims](#)

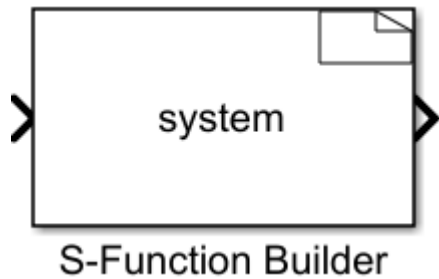


MathWorks Virtual Vehicle: C code integration



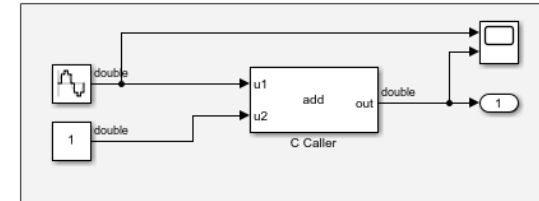
Integrate controller algorithms:

- Native Simulink models
- 3rd party tools (S-function, FMU, ...)
- C / C++ code



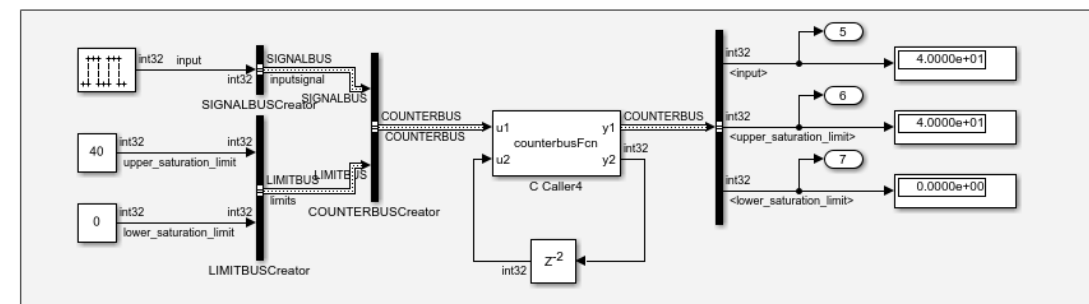
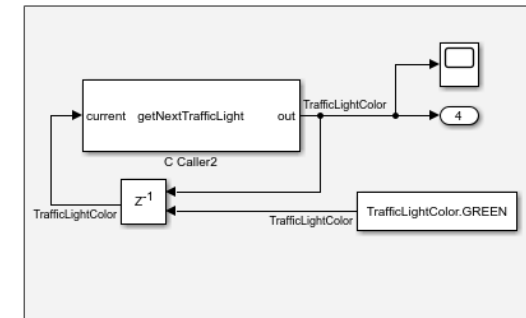
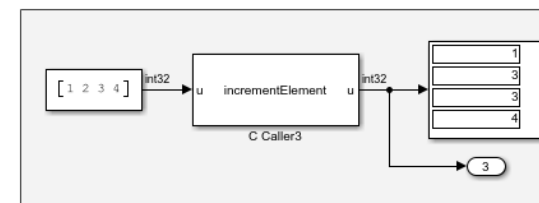
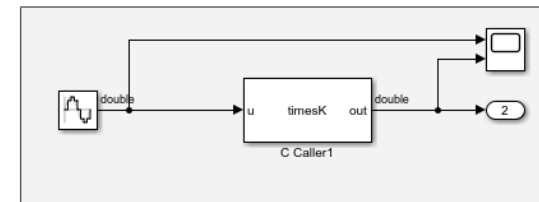
```
typedef struct {
    double coeff;
    double init;
    fault_T fault;
} params_T;
```

Name	Data Type
coeff	double
init	double
fault	Enum: fault_T



Call C Functions Using C Caller Block

```
matlabroot\toolbox\simulink\simdemos\simfeatures\include\my_func.h
matlabroot\toolbox\simulink\simdemos\simfeatures\src\my_func.c
```



Learn more:

[C / C++ code integration](#)

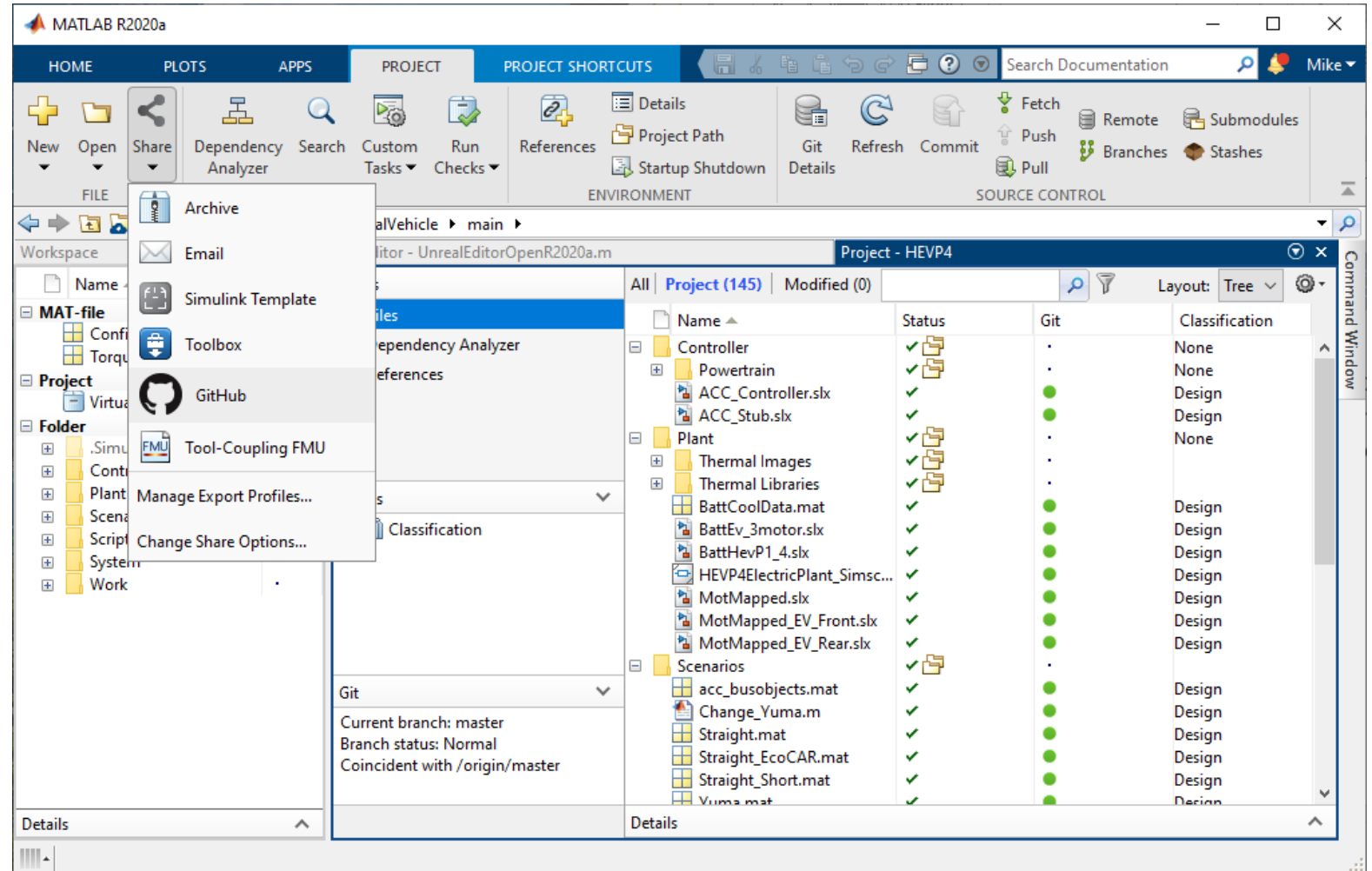
[C Caller block](#)

MathWorks Virtual Vehicle: complex project management



Use MathWorks platform to:

- Collaborate across teams
- Reference related project files
- Manage version control



Learn more:

[MATLAB Projects](#)

MathWorks Virtual Vehicle: graphical scenario authoring

Create
Vehicle

Integrate
Software

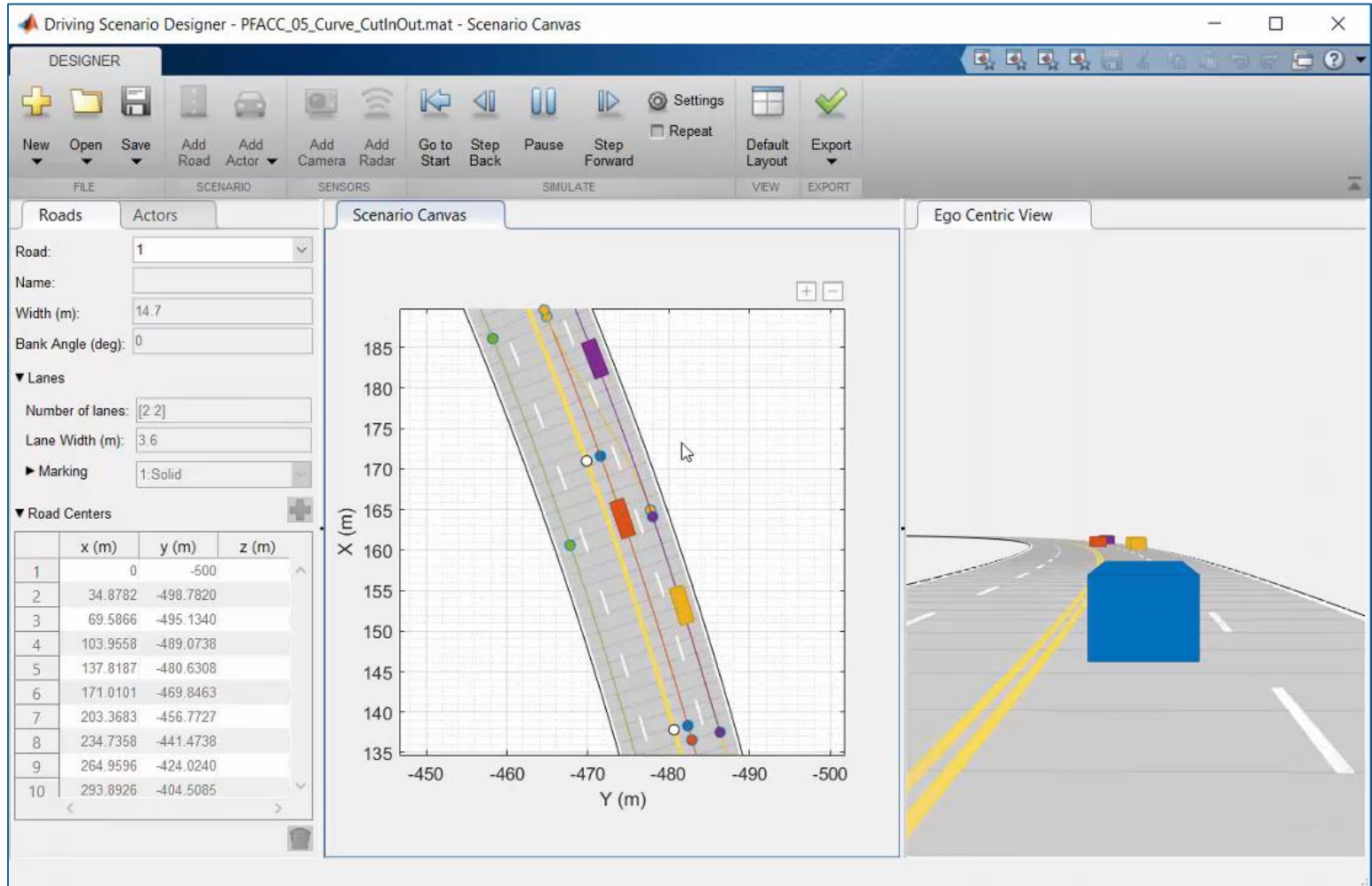
Author
Scenarios

Simulate &
Analyze

Deploy
Simulation

Use Driving Scenario Designer to:

- Create roads and lane markings
- Add actors and trajectories
- Specify actor size and radar cross-section (RCS)
- Explore pre-built scenarios
- Import OpenDRIVE and HERE HD Live Map roads
- Export MATLAB code
- Export Simulink model



Learn more:

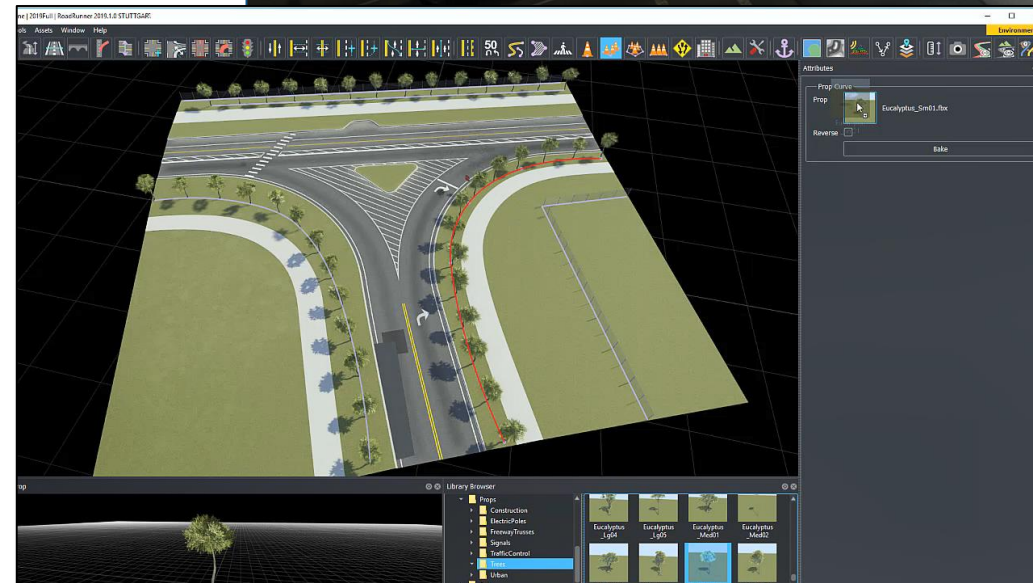
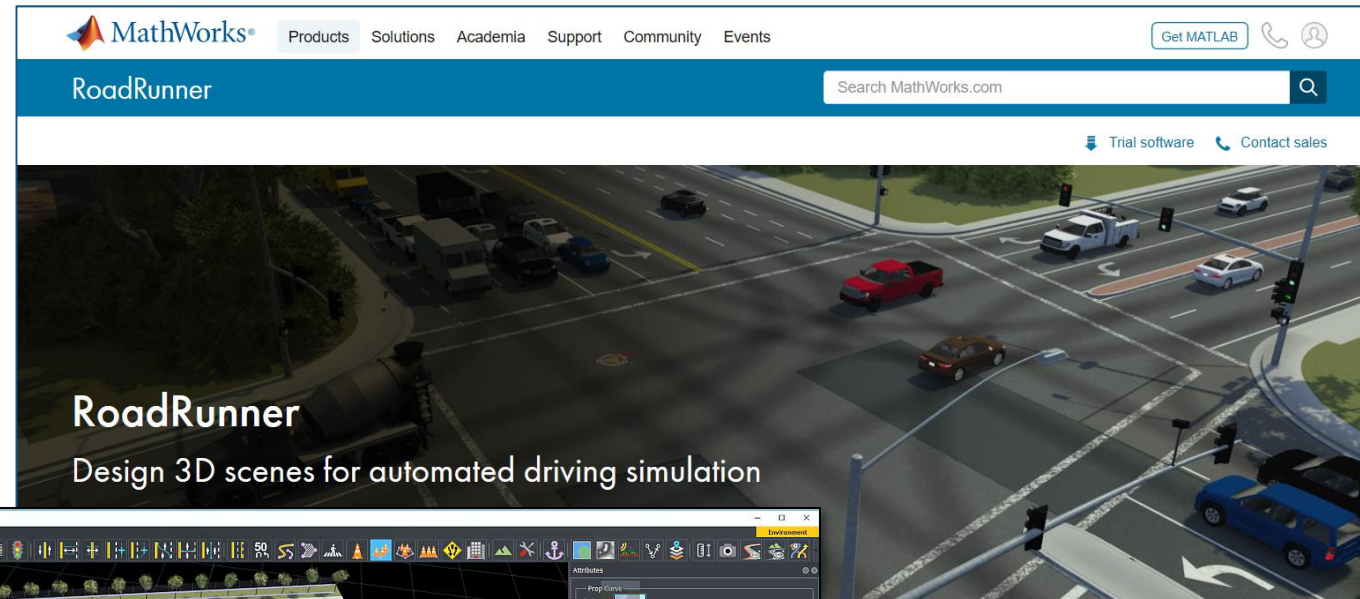
[Automated Driving Toolbox](#)

MathWorks Virtual Vehicle: automotive scene creation



Use RoadRunner to:

- Design 3D scenes for AD simulation
- Customize with region-specific road signs and markings
- Configure traffic signal timing
- Import from OpenDRIVE
- Export to OpenDRIVE, FBX, ...
- Use scenes in Unreal, Unity, CARLA, ...



Learn more:

[RoadRunner](#)

MathWorks Virtual Vehicle: requirements definition



Use V&V tools to:

- Define sequence of simulations to run
- Define requirements for these tests
- Define custom report template

HEVP2

Test Result Information

Result Type: Test File Result
 Parent: None
 Start Time: 04-Mar-2019 07:25:34
 End Time: 04-Mar-2019 07:43:23
 Outcome: Total: 8, Passed: 6, Failed: 2

Test Suite Information

Name: HEVP2
[Back to Report Summary](#)

Performance

Test Result Information

Result Type: Test Suite Result
 Parent: [HEVP2](#)
 Start Time: 04-Mar-2019 07:25:34
 End Time: 04-Mar-2019 07:33:16
 Outcome: Total: 2, Passed: 2

Test Suite Information

Name: Performance
[Back to Report Summary](#)

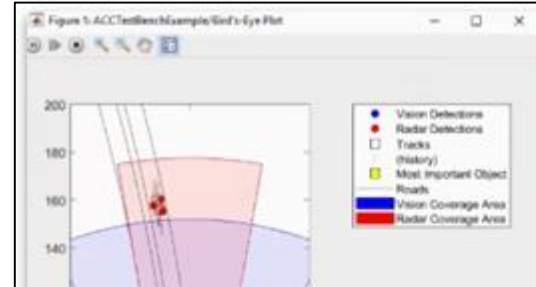
Learn more:
[Verification & Validation](#)

MathWorks Virtual Vehicle: results analysis



Use post-processing tools to:

- Review results with flexible MATLAB platform and visualization tools
- Interact with user-friendly Live Scripts
- Automate report generation



Live Editor window showing MATLAB code for report generation:

```

gensdd.mlx *
LIVE EDITOR  INSERT  VIEW
gensdd.mlx +
164
165 finder = ChartDiagramFinder(hModel);
166 charts = find(finder);
167 ch = Chapter("Title", "Charts");
168 for chart = charts
169     section = Section("Title", chart.Name);
170     diag = getReporter(chart);
171     diag.SnapshotFormat = getSnapshotFormat(rpt);
172     add(section, diag);
173
174 % Report the objects in this chart
175 objFinder = StateflowDiagramElementFinder(chart);
176 sfObjects = find(objFinder);
177 for sfObj = sfObjects
178     objSection = Section("Title", sfObj.Name);
179     add(objSection, sfObj);
180     add(section, objSection);
181 end
182 add(ch, section);
183 end
184 add(rpt, ch);
185
gensdd Ln 190 Col 11
    
```

Below the code is a Simulink block diagram for 'gensdd' showing components like 'Input Processor', 'Engine', 'Transmission', and 'Vehicle Power'.

Report window titled 'sdd_sf_car.pdf' showing 'Chapter 2. Charts' and '2.1.1. shift_logic'.

Figure 2.1. shift_logic

2.1.1. gear_state

Table 2.1. sf_car/shift_logic/gear_state Properties

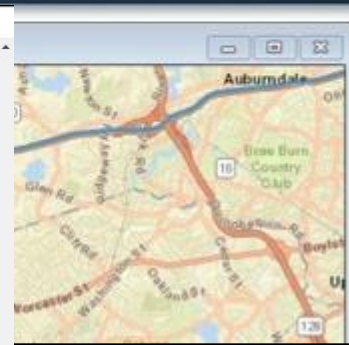
Property	Value
Type	AND State
Label	gear_state
Events	DOWN UP

2.1.2. selection_state

Table 2.2. sf_car/shift_logic/selection_state Properties

Property	Value
Type	AND State
Label	selection_state
during	{down_th,up_th} = calc_th(gear,throttle);

36



Learn more:

[MATLAB Live Editor](#)

[Simulink Report Generator](#)

MathWorks Virtual Vehicle: scalability

Create
Vehicle

Integrate
Software

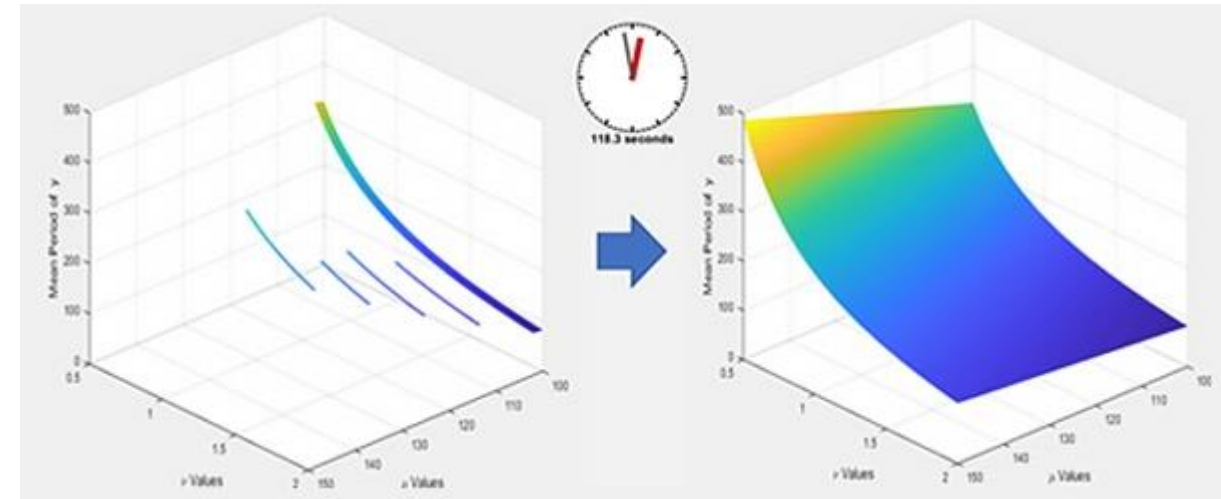
Author
Scenarios

Simulate &
Analyze

Deploy
Simulation

Use MATLAB and Simulink to:

- Distribute simulations to local multi-core, GPU, clusters, or the cloud
- Scale up computation power as needed without needing to rewrite code



Learn more:

[Parallel Computing Toolbox](#)

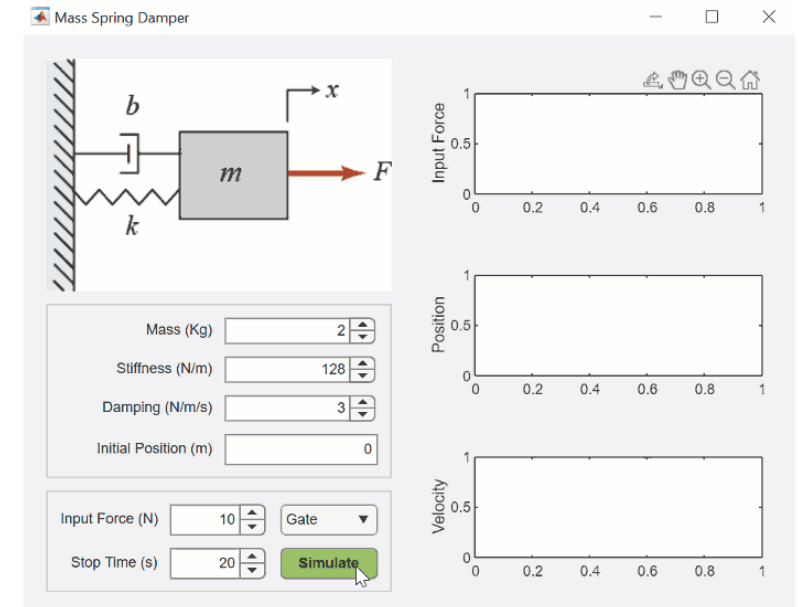
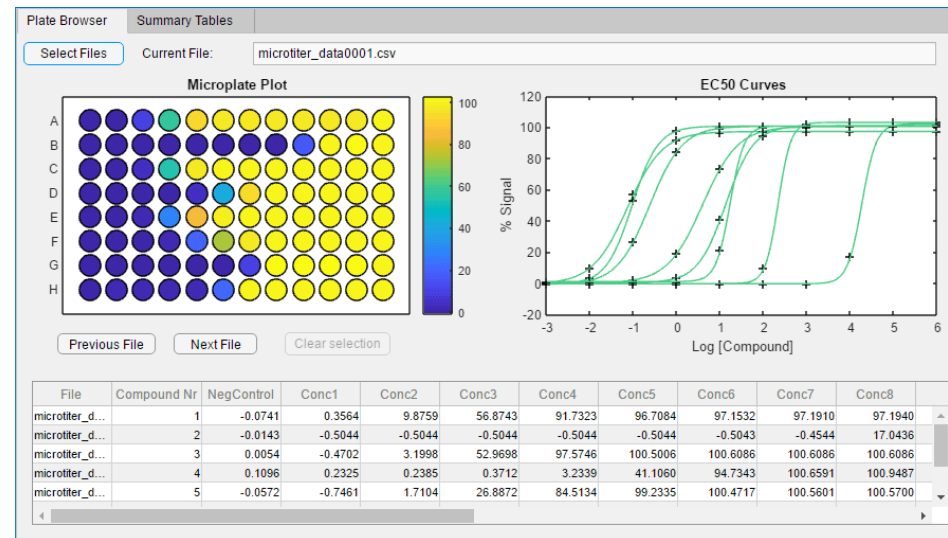
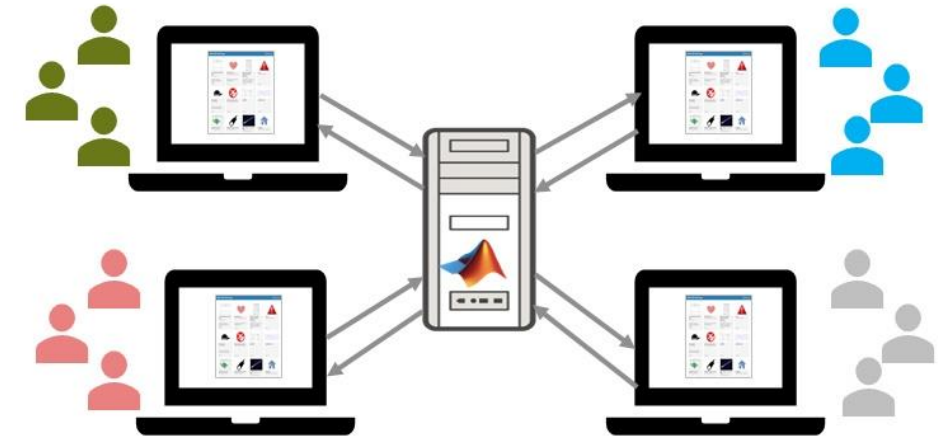
[MATLAB Parallel Server](#)

MathWorks Virtual Vehicle: model deployment



Use MATLAB and Simulink to take applications farther:

- Create custom UI's
- Create installers for distribution
- Deploy models as executables, FMU's or web apps
- Generate code for SIL, HIL testing



Learn more:

[MATLAB Web App Server](#)

[MATLAB App Designer](#)

[Simulink Compiler](#)

[Embedded Systems](#)

MathWorks Consulting Services can support you



Model Architecture

Model assessment
Simulation performance
Interface standardization
...



Construction

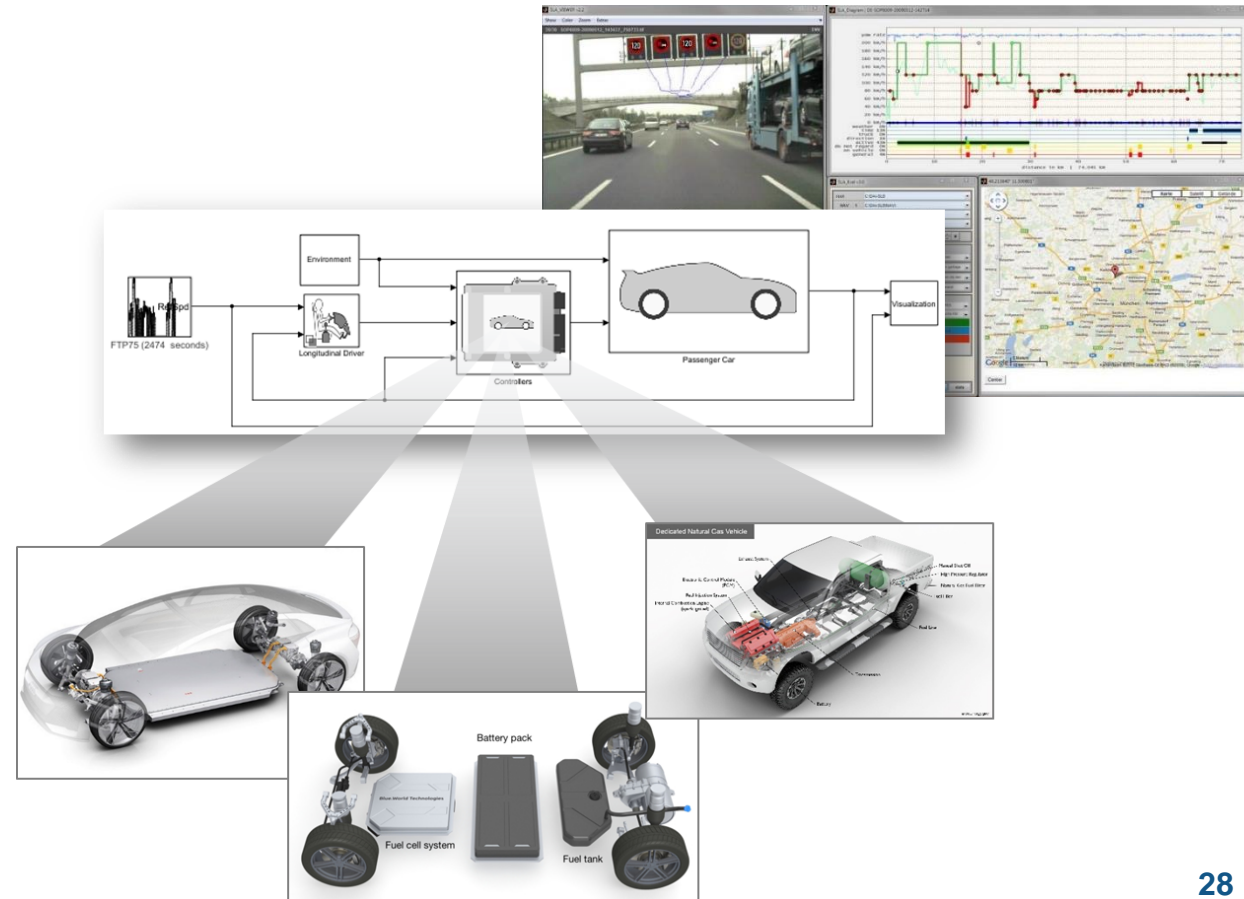
Build process automation
Database/Repo interface
Model-Building know-how
...



User Experience

GUI driven workflow
Tool compatibility support
Artifact creation
...

- Provide expert-level guidance
- Automate workflows
- Develop custom UI's



Agenda

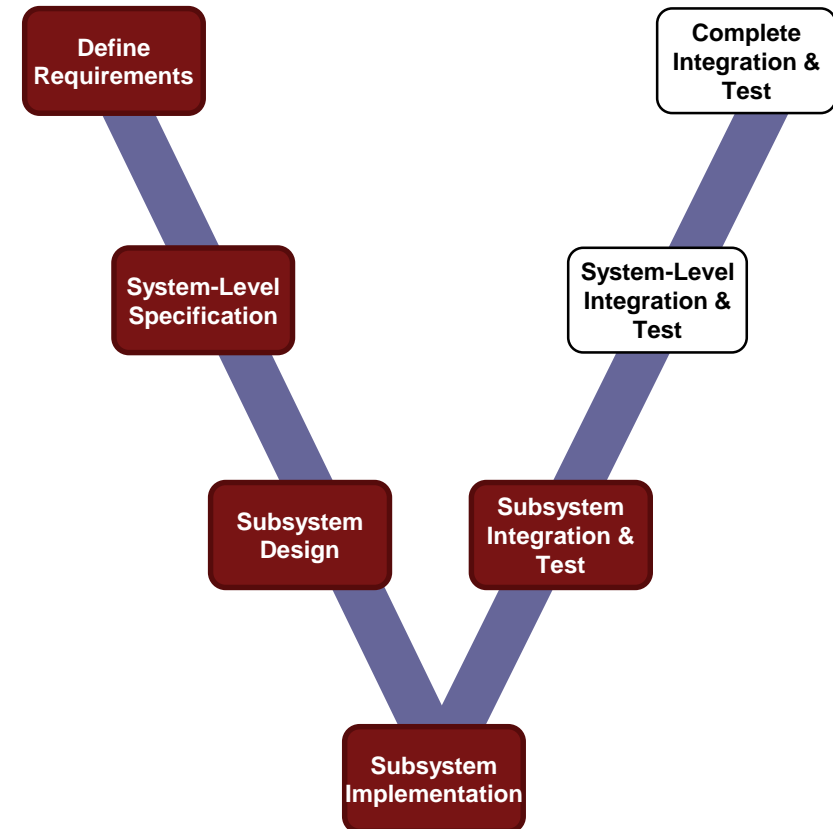
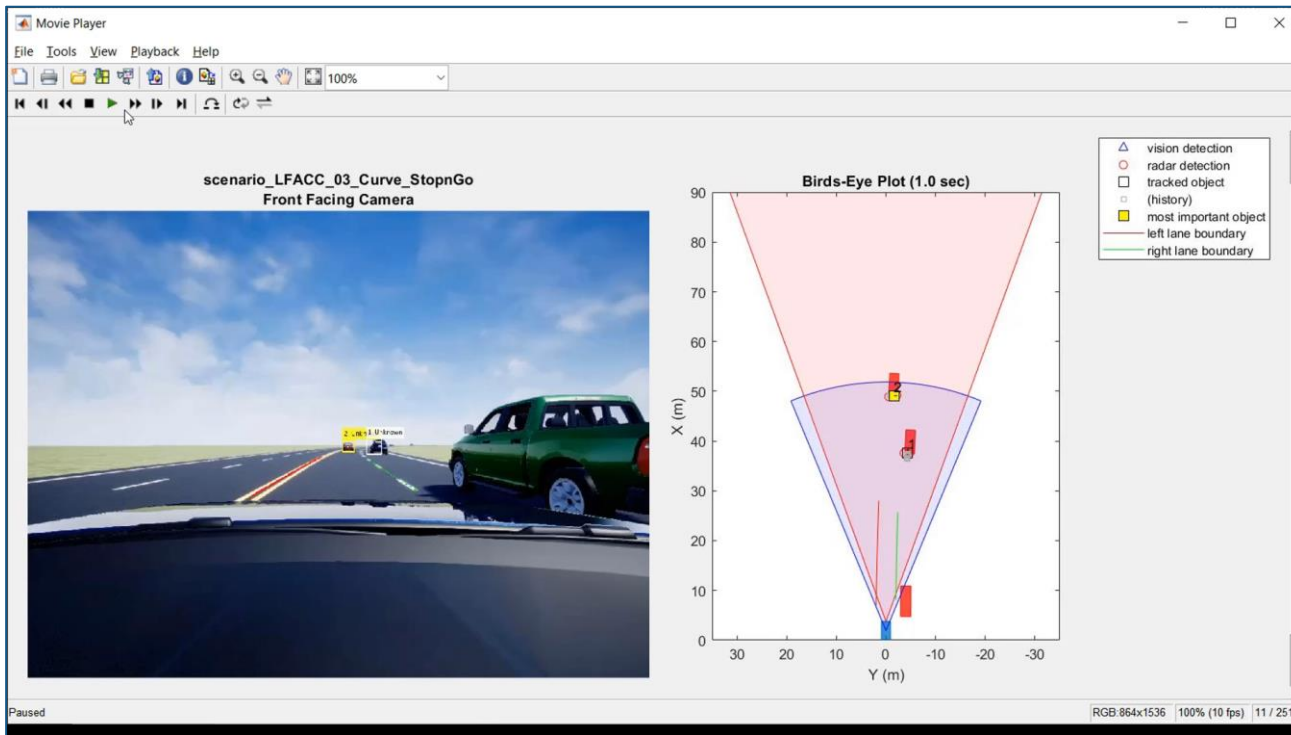
- Common challenges
- MathWorks solutions
- **Case study**

Validate software against function safety requirements early



FSR: The lane following system lateral error shall be less than 1 meter

*Use simulation to do system-level integration testing **early***

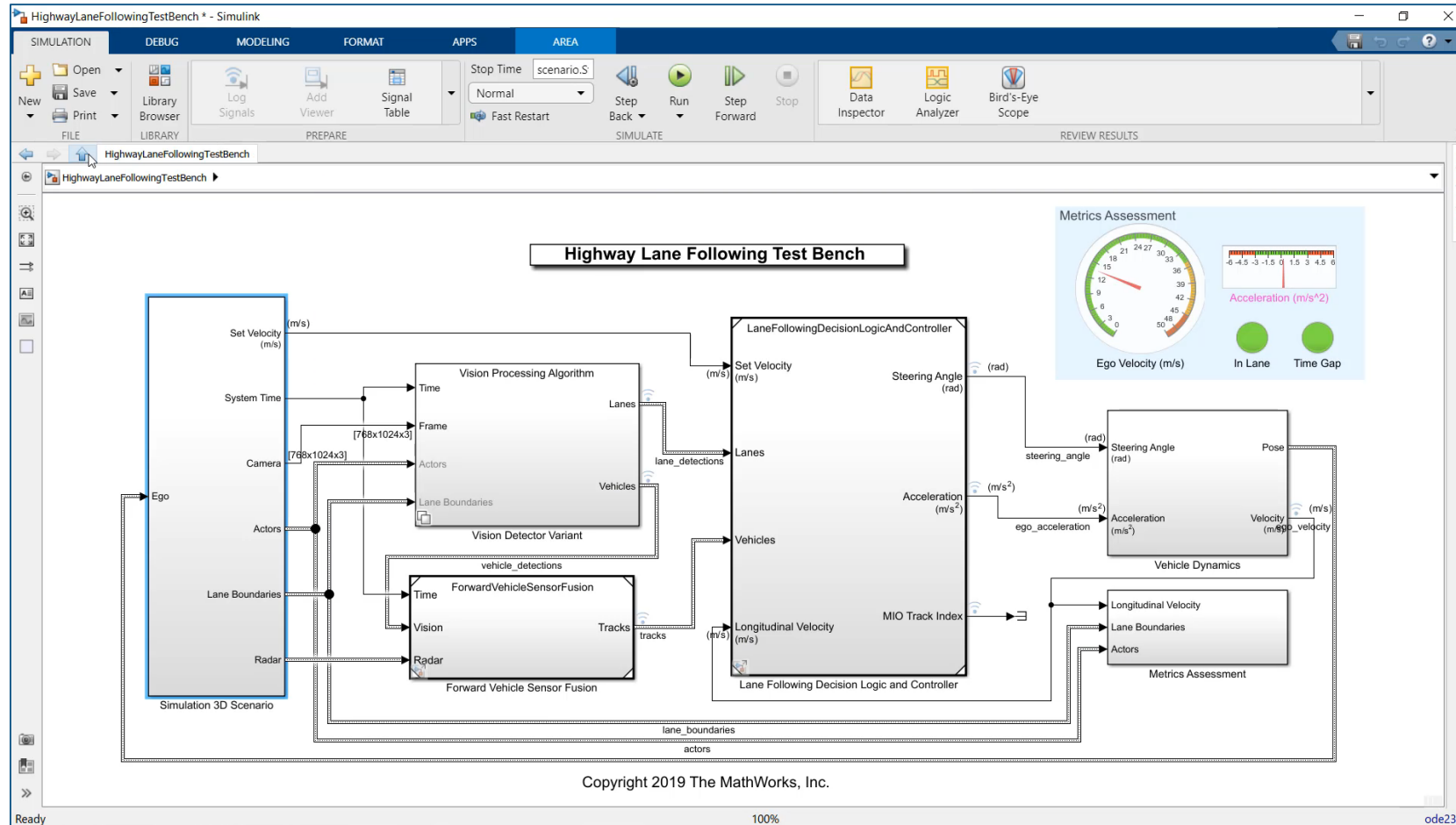


Learn more:

[Highway Lane Following](#)

[Automate Testing for Highway Lane Following](#)

Case study: highway lane following algorithm



- Create Unreal Engine scene
- Specify target trajectories
- Model camera and radar sensors
- Model ego vehicle dynamics
- Specify system metrics

Learn more:
[Highway Lane Following](#)

Case study: highway lane following algorithm



The screenshot shows the Requirements Editor interface. On the left, a tree view lists requirements under 'HighwayLaneFollowingTestRequirements'. Requirement #3, 'scenario_LFACC_03_Curve_StopnGo', is selected. The main area displays the details for this requirement, including its type (Functional), index (3), and summary. A table provides a detailed description of the test scenario, including host car, lead car, and third car parameters.

Test Description	Host Car	Lead Car	Third Car
Stop and Go in Curved highway	initial velocity = 14 m/s HW = 50m v_set = 14 m/s	initial velocity = 14 m/s Lead car slows down to 8m/s and stay constant for 10s, then speed up to 13 m/s.	2 slow moving cars at 8 m/s in 3 rd lane 2 fast Moving cars at 15 m/s in the 1 st lane

- Author and associate requirements and scenarios

Learn more:

[Automate Testing for Highway Lane Following](#) 32

Case study: highway lane following algorithm



The image displays the simulation environment for a highway lane following algorithm. On the left, the Simulink model 'Highway Lane Following Test Bench' is shown, featuring blocks for sensor fusion (Camera, Radar, Vision), lane detection, vehicle dynamics, and a control controller. A 'Metrics Assessment' gauge is visible. The top right shows a 3D Unreal Engine view of a blue car following a green truck on a highway. The bottom left shows a 'Floating Scope' window with two plots: 'Velocity' (m/s) and 'Relative Distance' (m), with a 4x magnification factor.

- Visualize system behavior with Unreal Engine
- Visualize lane detections
- Visualize vehicle detections
- Visualize control signals
- Log simulation data

Learn more:
[Highway Lane Following](#)

Case study: highway lane following algorithm



The screenshot displays the Test Manager interface for a simulation test. The main window shows a block diagram of the 'Highway Lane Following Test Bench' with various components like 'Vision Processing Algorithm', 'Lane Following Decision Logic and Controller', and 'Vehicle Dynamics'. A 'Metrics Assessment' panel on the right shows gauges for 'Ego Velocity (m/s)' and 'Acceleration (m/s²)'. The left sidebar lists several test scenarios, with 'scenario_LFACC_03_Curve_StopnGo' selected. The bottom left shows a 'PROPERTY' table for the selected test.

PROPERTY	VALUE
Name	scenario_LFACC_03_Curve_StopnGo
Type	Simulation Test
Model	HighwayLaneFollowingTestBench
Simulation Mode	[Model Settings]
Location	D:\userpath\work\Examples\R2020...
Enabled	<input checked="" type="checkbox"/>
Hierarchy	HighwayLaneFollowingMetricAsses...
Tags	Type comma or space separated tags

- Automate test execution and reporting
- Execute simulations in parallel

Learn more:

[Automate Testing for Highway Lane Following](#) 34

Case study: highway lane following algorithm



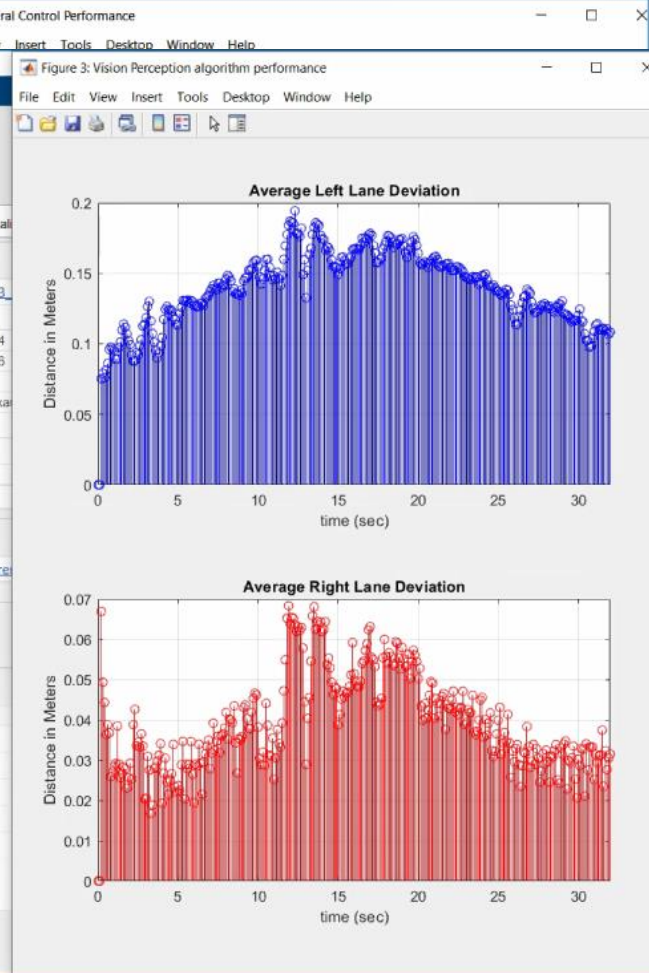
The screenshot displays the Test Manager interface with the following components:

- TESTS Panel:** A tree view showing test results for 'HighwayLaneFollowingMetricAssessments' and 'Test Scenarios'. The selected scenario is 'scenario_LFACC_03_Curve_StopnGo' with a status of '10' and a green checkmark.
- Properties Panel:** A table showing simulation metadata:

Name	scenario_LFACC_03_Curve_StopnGo
Outcome	10
Start Time	03/23/2020 17:53:14
End Time	03/23/2020 17:56:26
Type	Simulation Test
Test File Location	D:\userpath\work\Exa...
Test Case Definition	
Rerun Test Case	
Tags	
Simulation Metadata	
- Sim Output (HighwayLaneFollowing):** A list of simulation metrics with status indicators:
 - relative_distance: green bar
 - relative_velocity: pink bar
 - mio: red bar
 - MPC Controller:1: green bar
 - lateral_deviation: blue bar
 - relative_yaw_angle: orange bar
 - Watchdog Braking Co...: yellow bar
- PROPERTY VALUE Table:**

PROPERTY	VALUE
Name	scenario_LFACC_03_Curve_StopnGo
Status	10
Start Time	03/23/2020 17:53:14
End Time	03/23/2020 17:56:26
Type	Simulation Test
Test File Location	D:\userpath\work\Examples\R2020...
Model	HighwayLaneFollowingTestBench
Simulation Mode	normal
Test Case Definition	

- Assess system metrics
- Assess lane detection metrics



Learn more:

[Automate Testing for Highway Lane Following](#) 35

Case study: highway lane following algorithm



The screenshot shows the Simulink C++ Code Editor interface. On the left, a Simulink block diagram titled "Lane Following Decision Logic and Controller" is visible. It includes blocks for "Lane Detectors", "Lane Center", "Estimate Lane Center", "Find Lead Car", and "Lane Following Controller". On the right, the C++ code editor displays the header file "LaneFollowingDecisionLogicAndController.h".

```

56 LaneFollowingDecisionLogicAndController_GetCAPIStaticMap
57
58 // Class declaration for model LaneFollowingDecisionLogicAndController
59 class PathFollowingControllerRefMdlModelClass {
60 // public data and function members
61 public:
62 // Block signals and states (default storage) for system
63 typedef struct {
64     real32_T Delay_4_DSTATE; // '<S1>/Delay'
65     real32_T Delay_3_DSTATE; // '<S1>/Delay'
66     real32_T Delay_2_DSTATE; // '<S1>/Delay'
67     real32_T Delay_1_DSTATE; // '<S1>/Delay'
68 } lfdlacDW_EstimateLaneCenter_T;
69
70 // Block signals and states (default storage) for system
71 typedef struct {
72     lfdlacDW_EstimateLaneCenter_T EstimateLaneCenter; // '<
73     real_T LaneFollowingController_o1; // '<Root>/Lane Fol
74     real_T relative_distance; // '<Root>/Find Lea
75     real_T relative_velocity; // '<Root>/Find Lea
    
```

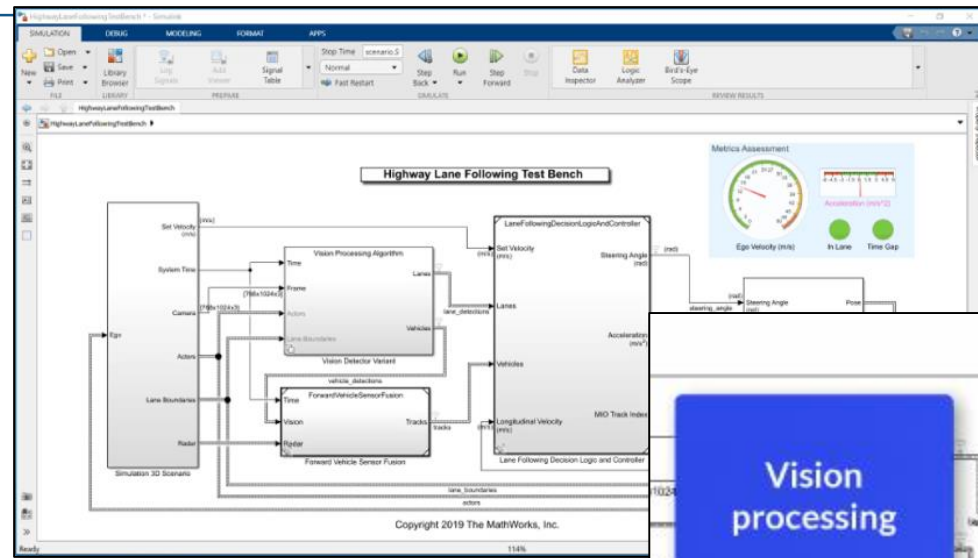
- Generate algorithm code
- Test with Software-in-the-Loop (SIL) simulation
- Workflow could be extended to test hand coded algorithms

Learn more:

[Automate Testing for Highway Lane Following](#) 36

Summary

1. Started with reference application, then customized
2. Integrated software
3. Defined scenarios to test
4. Simulated model and analyzed results
5. Deployed model



Vision processing

Controls and decision logic

File Edit Display Analysis Report Help

View: Requirements

Index	ID	Summary
1	#1	scenario_LFACC_01_Curve_DecelTarget
2	#2	scenario_LFACC_02_Curve_AutoRetarget
3	#3	scenario_LFACC_03_Curve_StopnGo
4	#4	scenario_LFACC_04_Curve_CutInOut
5	#5	scenario_LFACC_05_Curve_CutInOut_TooClose
6	#6	scenario_LFACC_06_Straight_StopandGoLeadC
7	#7	scenario_LF_01_Straight_RightLane
8	#8	scenario_LF_02_Straight_LeftLane
9	#9	scenario_LF_03_Curve_LeftLane
10	#10	scenario_LF_04_Curve_RightLane

Properties

Type: Functional

Index: 3

Custom ID: #3

Summary: scenario_LFACC_03_Curve_StopnGo

Description: Rationale

Test Description	Host Car	Lead Car	Third Car
Stop and Go in Curved highway	initial velocity = 14 m/s	initial velocity = 14 m/s	2 slow moving cars at 8 m/s in 3 rd lane

Block Parameters: Lane Marker Detector

Model Reference

Reference the specified model.

Main Instance parameters

Model name: LaneMarkerDetector.sbx

Simulation mode: Normal

Model events simu: Normal

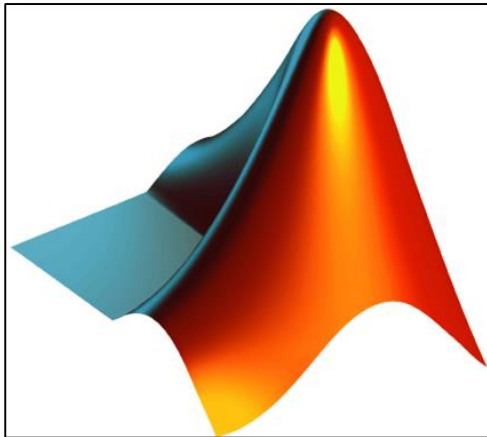
Show model ini: Software-in-the-loop (SIL)

Show model ter: Processor-in-the-loop (PIL)

Schedule rates

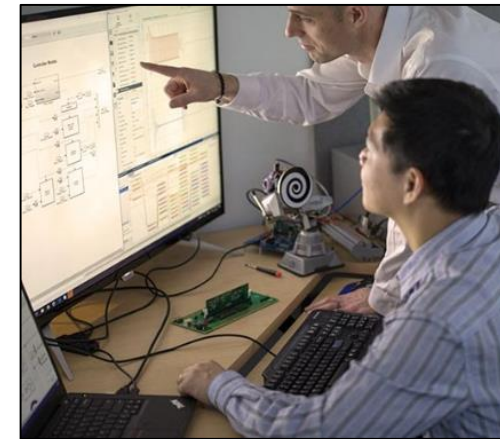
Key takeaways

MathWorks provides a **powerful platform** for building your **Virtual Vehicle**



Out-of-the-box capability

Our platform is very **flexible**, and we can help you **customize** it for your needs



Custom virtual vehicle solution



Presenter contact info and poll questions

Please contact us with questions



Chris Fillyaw
Application Engineering Manager
cfillyaw@mathworks.com



Mike Sasena, PhD
Automotive Product Manager
msasena@mathworks.com

- On a scale of 1 - 4, how challenging is it for your department to:
 - Create the vehicle model
 - Integrate software
 - Author scenarios
 - Simulate and analyze results
 - Deploy simulations

A. 1 (easy)

B. 2 (moderate)

C. 3 (difficult)

D. 4 (major challenge)

- Are you interested in a follow-up conversation with MathWorks?
- Additional comments

A. Yes

B. No

Thank You



Chris Fillyaw
Application Engineering Manager
cfillyaw@mathworks.com



Mike Sasena, PhD
Automotive Product Manager
msasena@mathworks.com