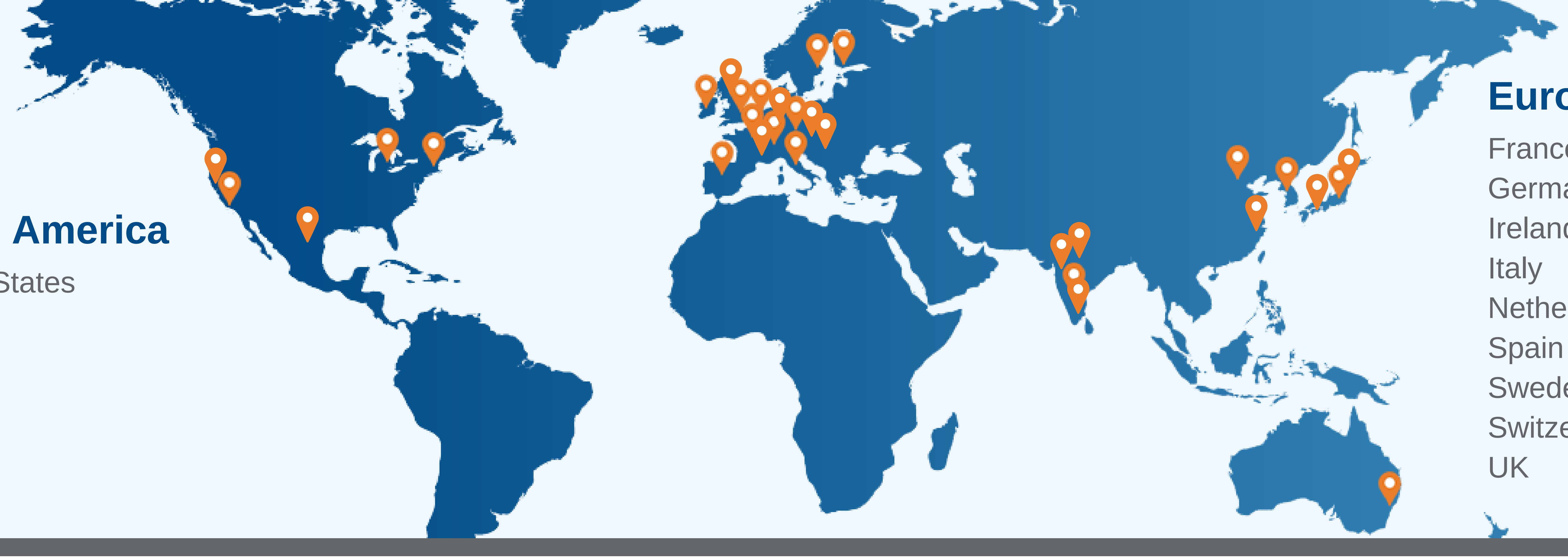
A look to the future with Model-Based Design

Andy Grace

Vice President of Engineering

Design Automation



Headquarters

Natick, MA USA

North America

United States

Europe

France
Germany
Ireland
Italy
Netherlands
Spain
Sweden
Switzerland
UK

Asia-Pacific

Australia
China
India
Japan
Korea

MathWorks Today

3 million+ users

in more than 180 countries

4500+ staff

in 31 offices around the world

$1B+

in 2018 revenues with 60% from outside the US

Privately held

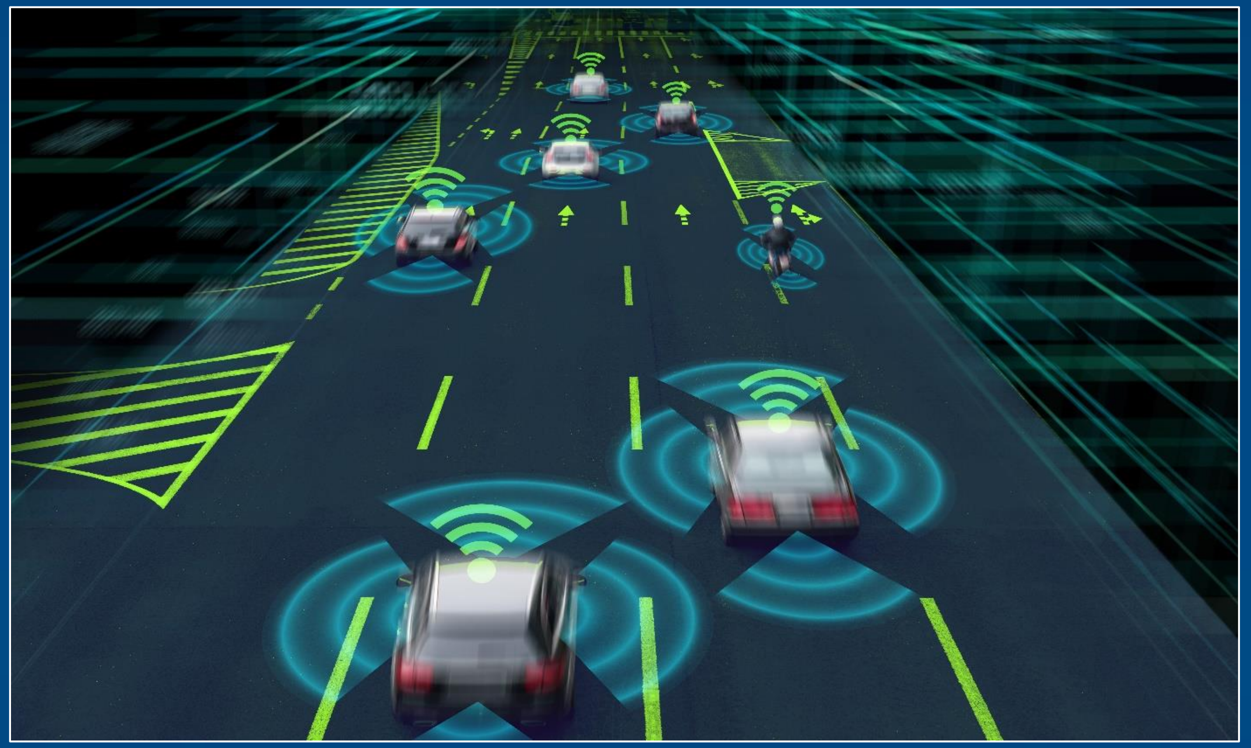
and profitable every year

MathWorks®

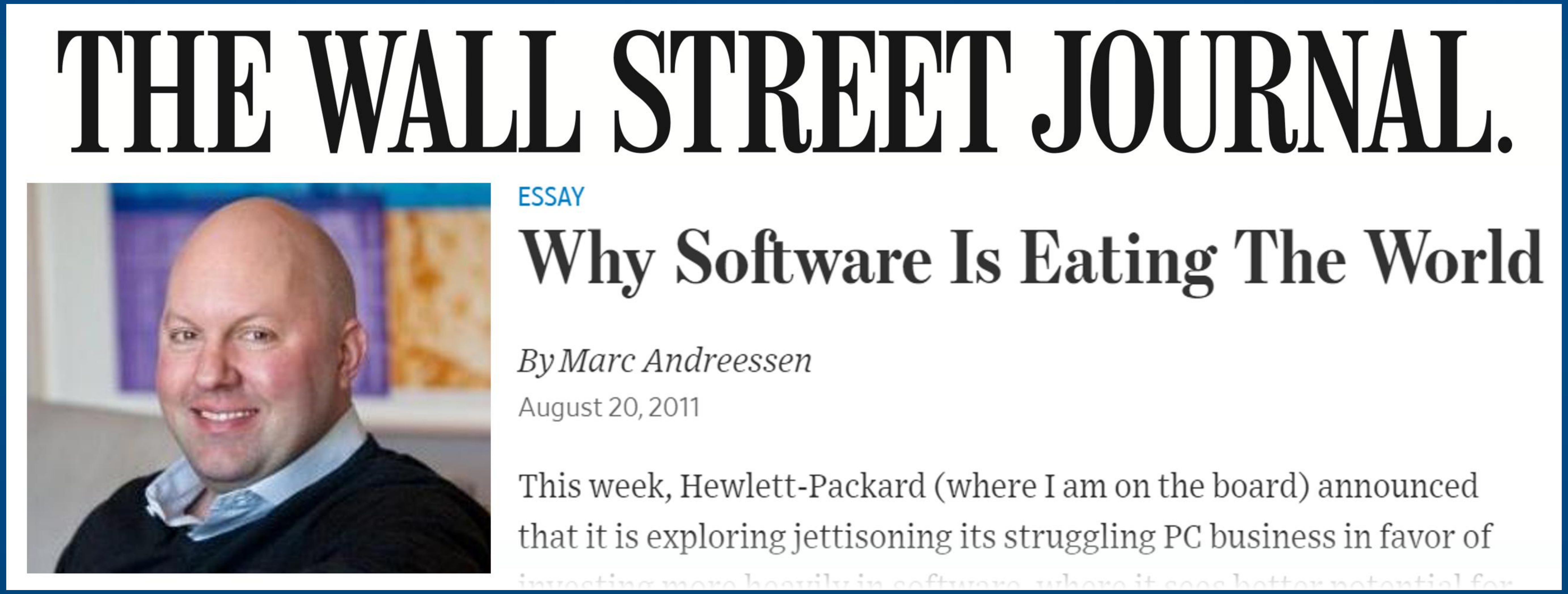
Technology Megatrends Driving Automotive

1. Vehicle Electrification
2. Autonomous Driving
3. Connected Vehicles

Software everywhere



Software is reshaping the automotive industry



THE WALL STREET JOURNAL.

ESSAY

Why Software Is Eating The World

By Marc Andreessen

August 20, 2011

This week, Hewlett-Packard (where I am on the board) announced that it is exploring jettisoning its struggling PC business in favor of investing more heavily in software, where it sees better potential for

In the future every company will become a <u>software</u> company

Marc Andreessen

Founder of Netscape,
Renowned Venture capitalist

Software is reshaping the automotive industry

Augmenting control with machine learning (BMW)

Trailer backup assist (Ford)

Autonomous driving (Voyage)

Agile Values

Individuals & Interactions over Process and Tools

Customer Collaboration over Contract Negotiation

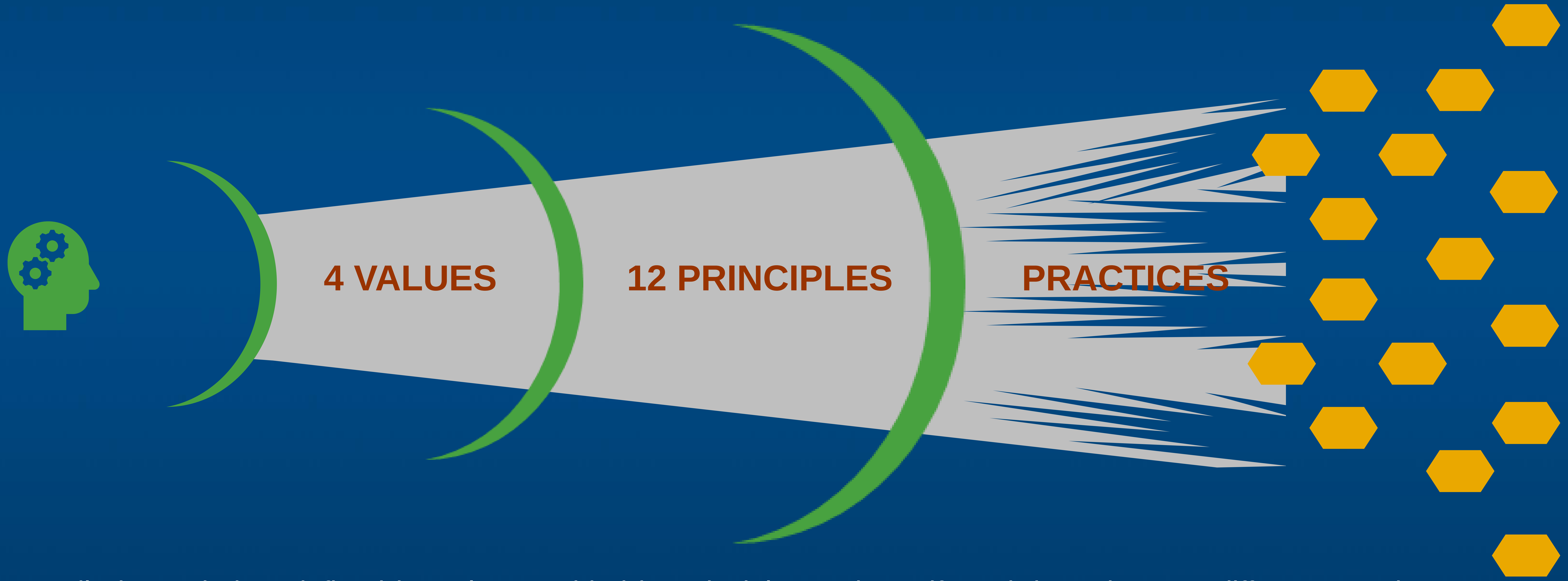
Working Software over Comprehensive Documentation

Responding to Change over Following a Plan

> “While there is value in the items on the right, we value the items on the left more.”

- The Agile Alliance, 2001

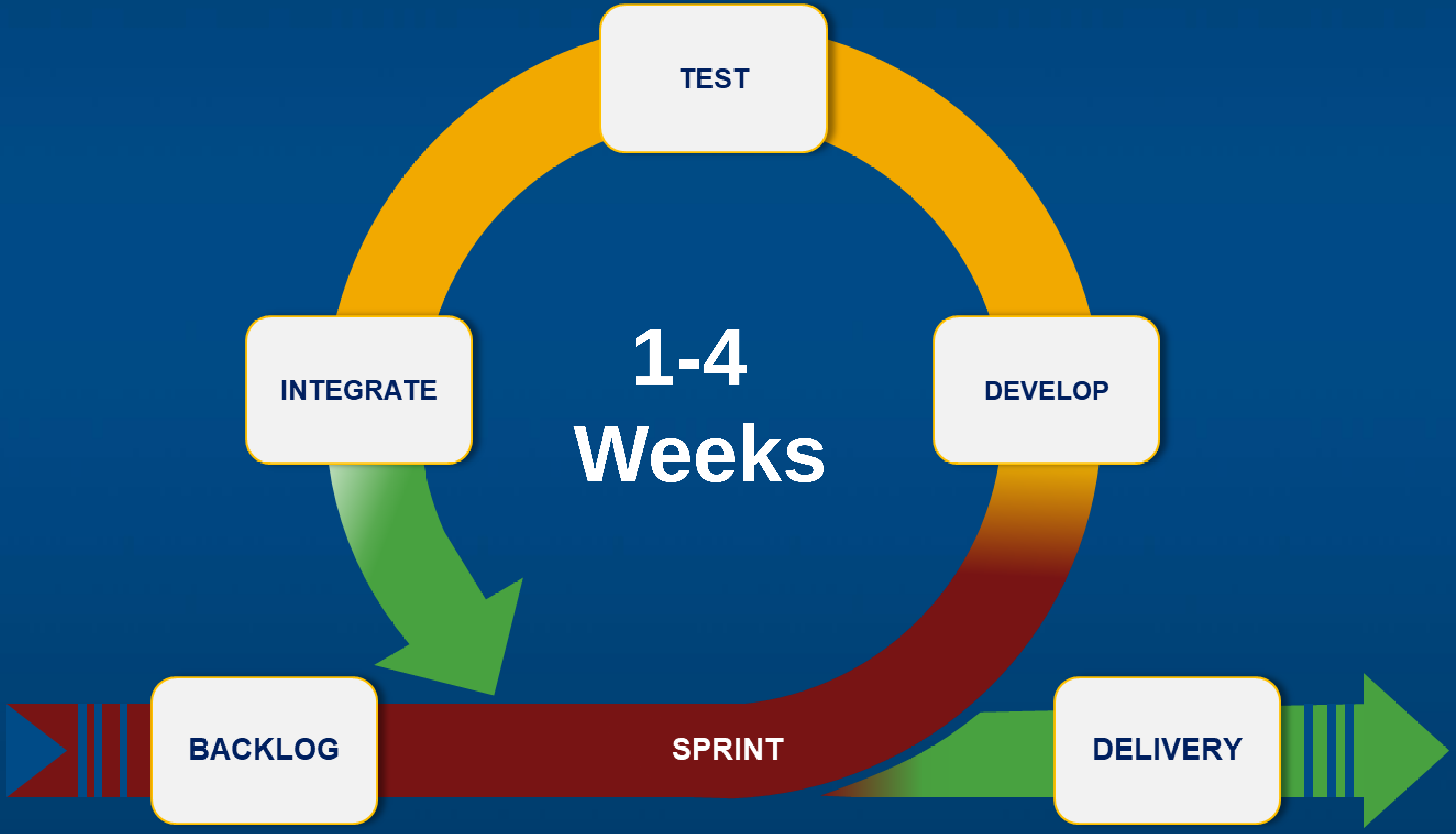
Agile: Values, Principles and Practices



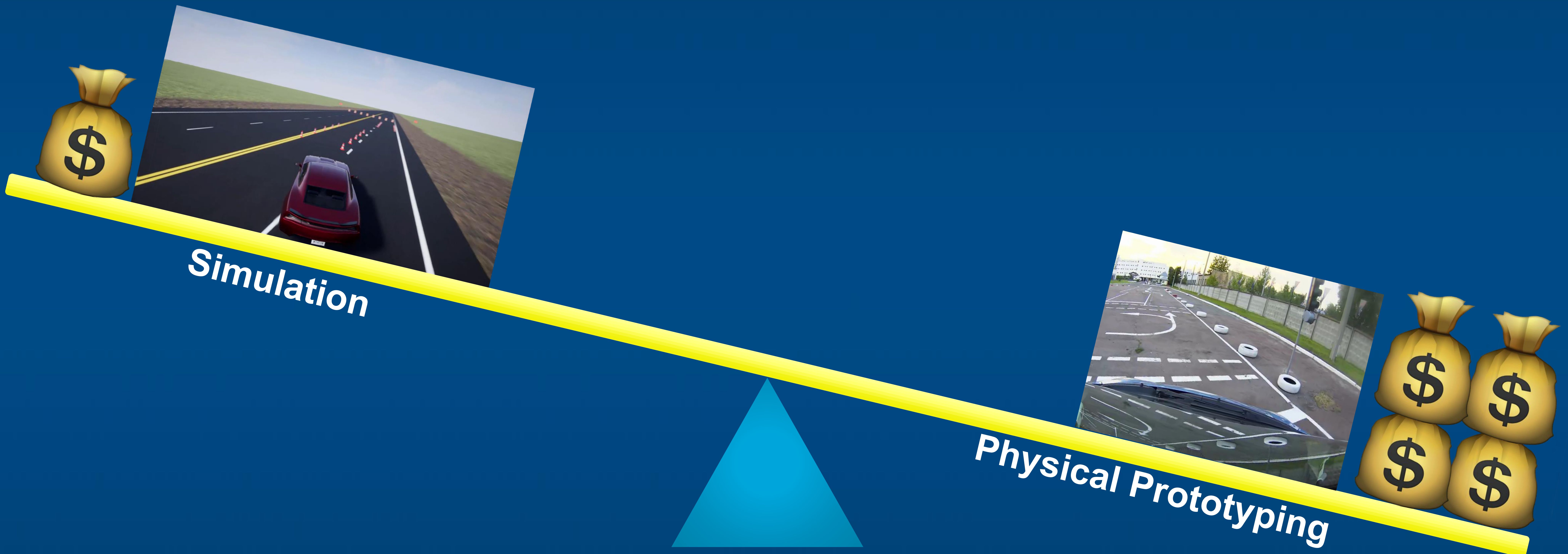
Agile is a mindset defined by values, guided by principles and manifested through many different practices. Agile practitioners select practices based on their needs.

~ Agile Practice Guide (PMI® and Agile Alliance®)

Typical agile development workflow

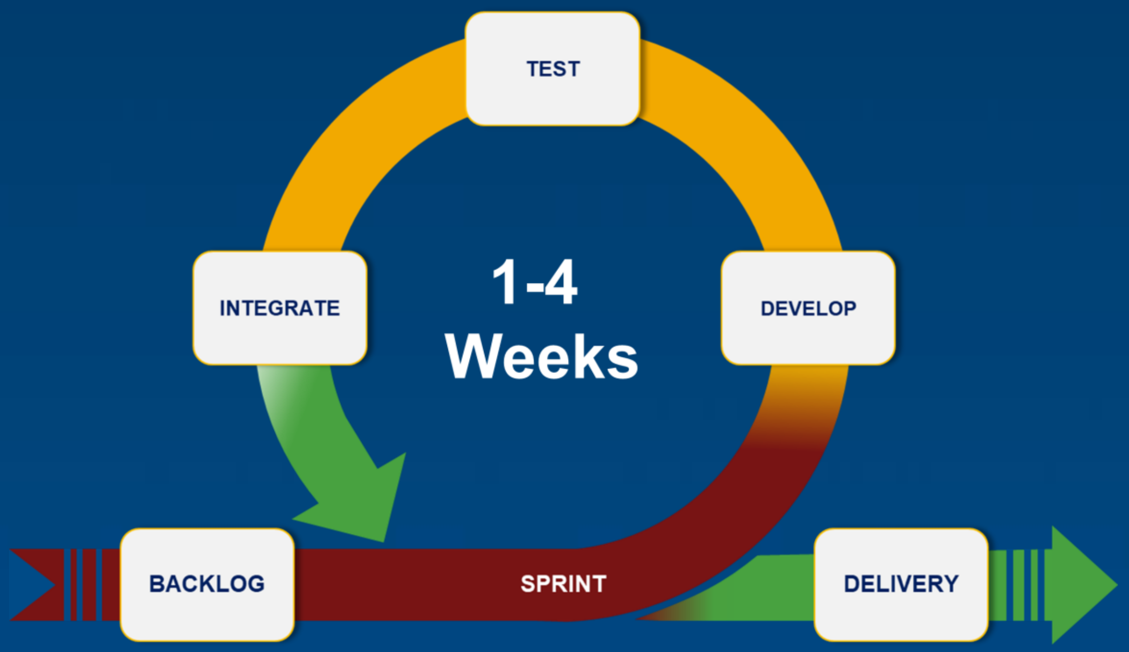


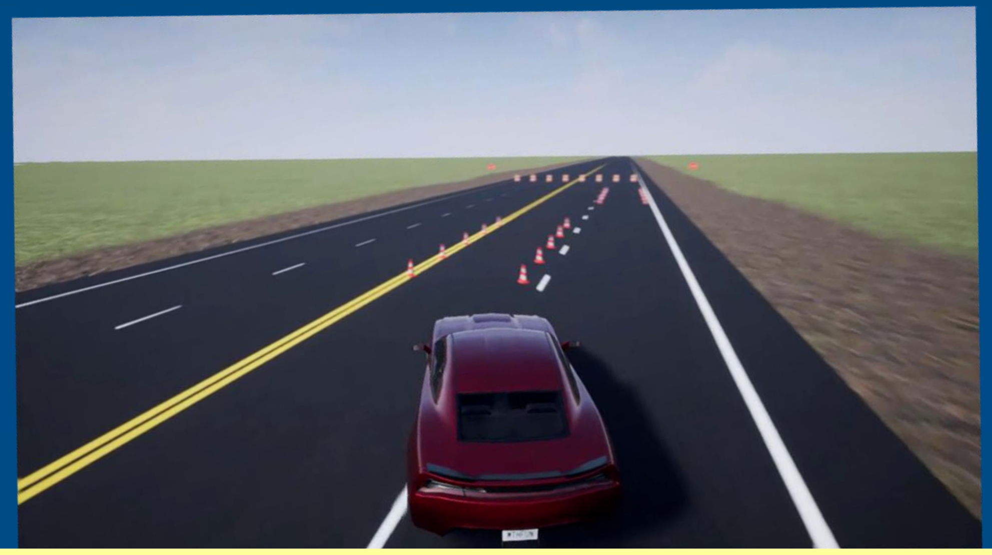
Models == Understanding



Simulation

Physical Prototyping

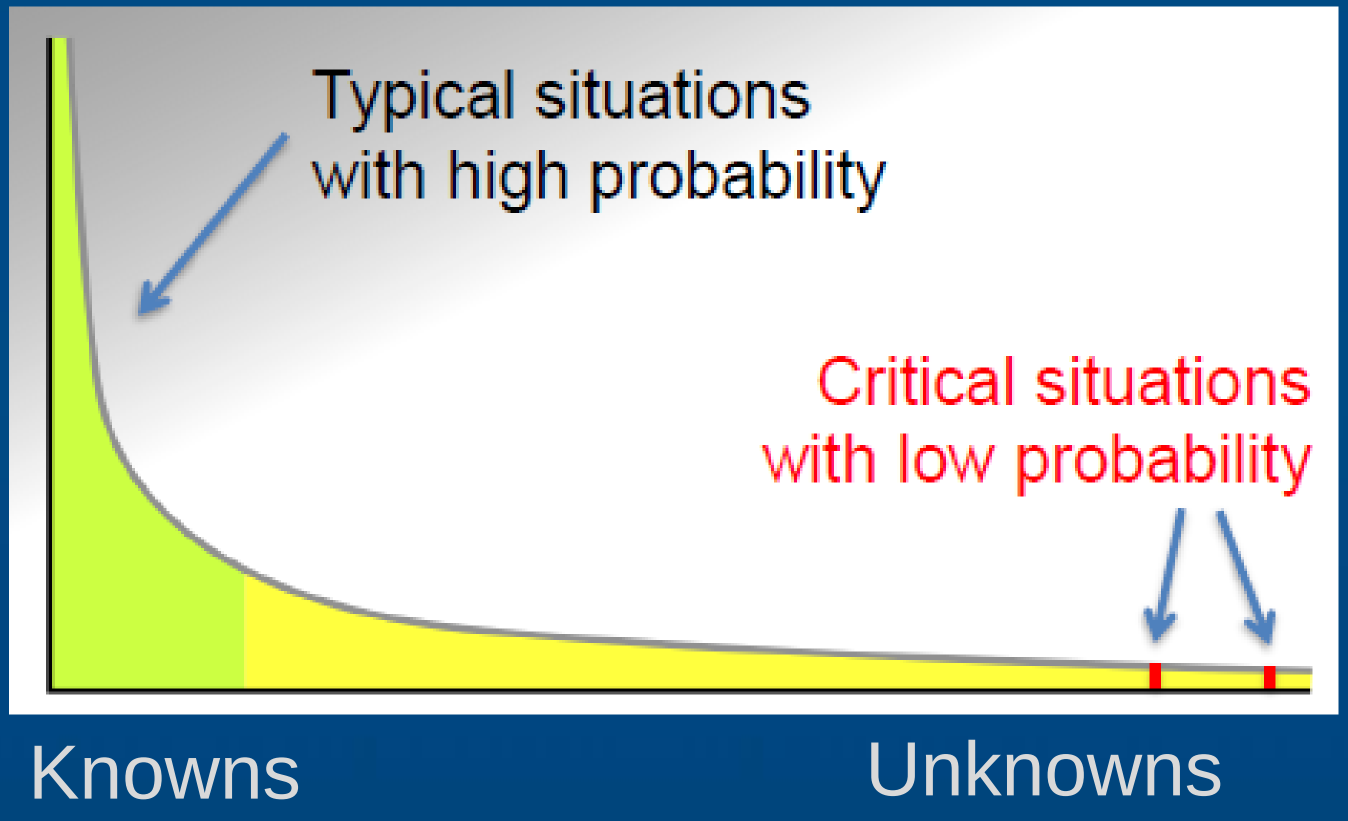




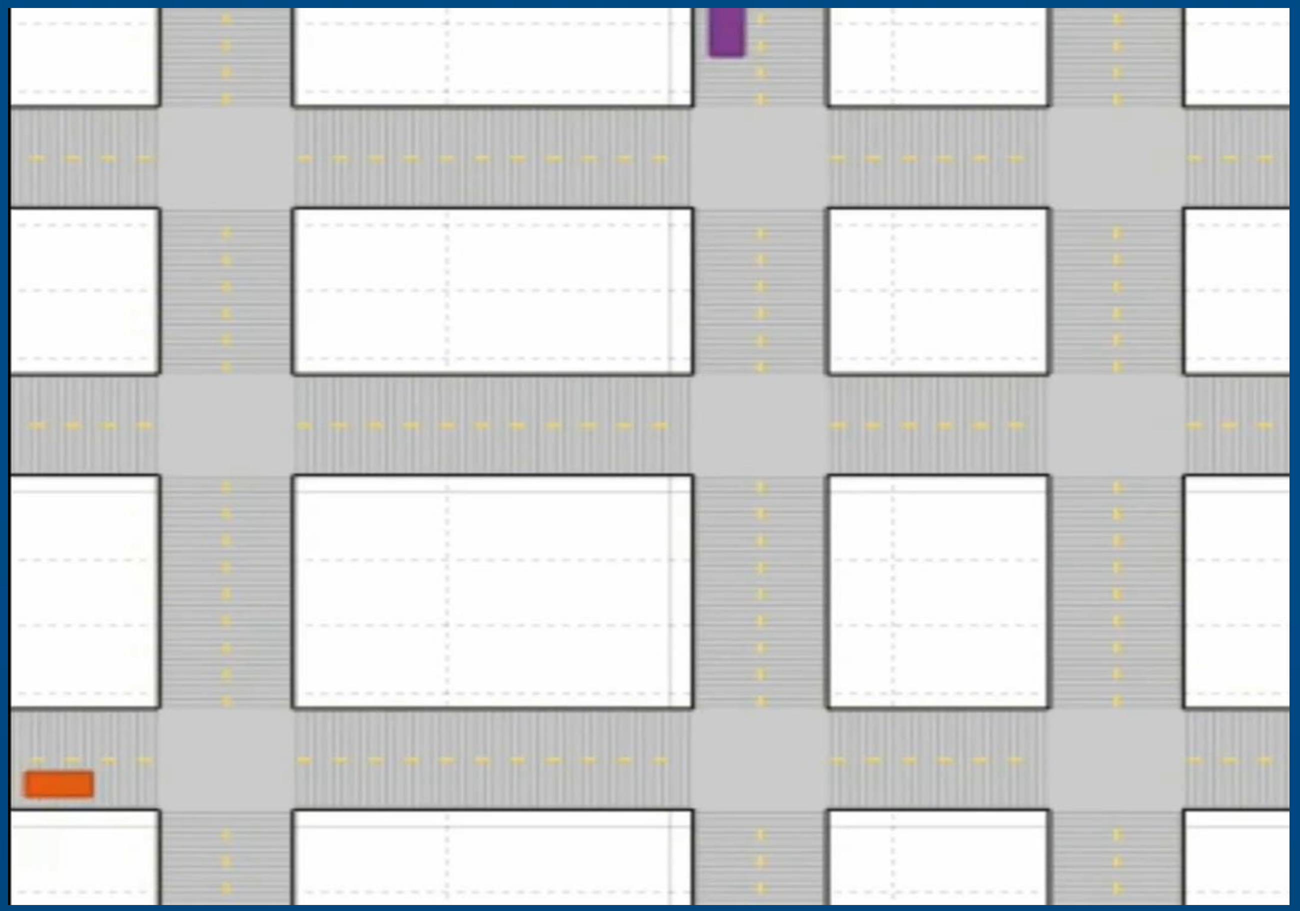
Simulation

Physical Prototyping

Simulation is key to Level 4-5 autonomy



Critical situations are in the long-tail*

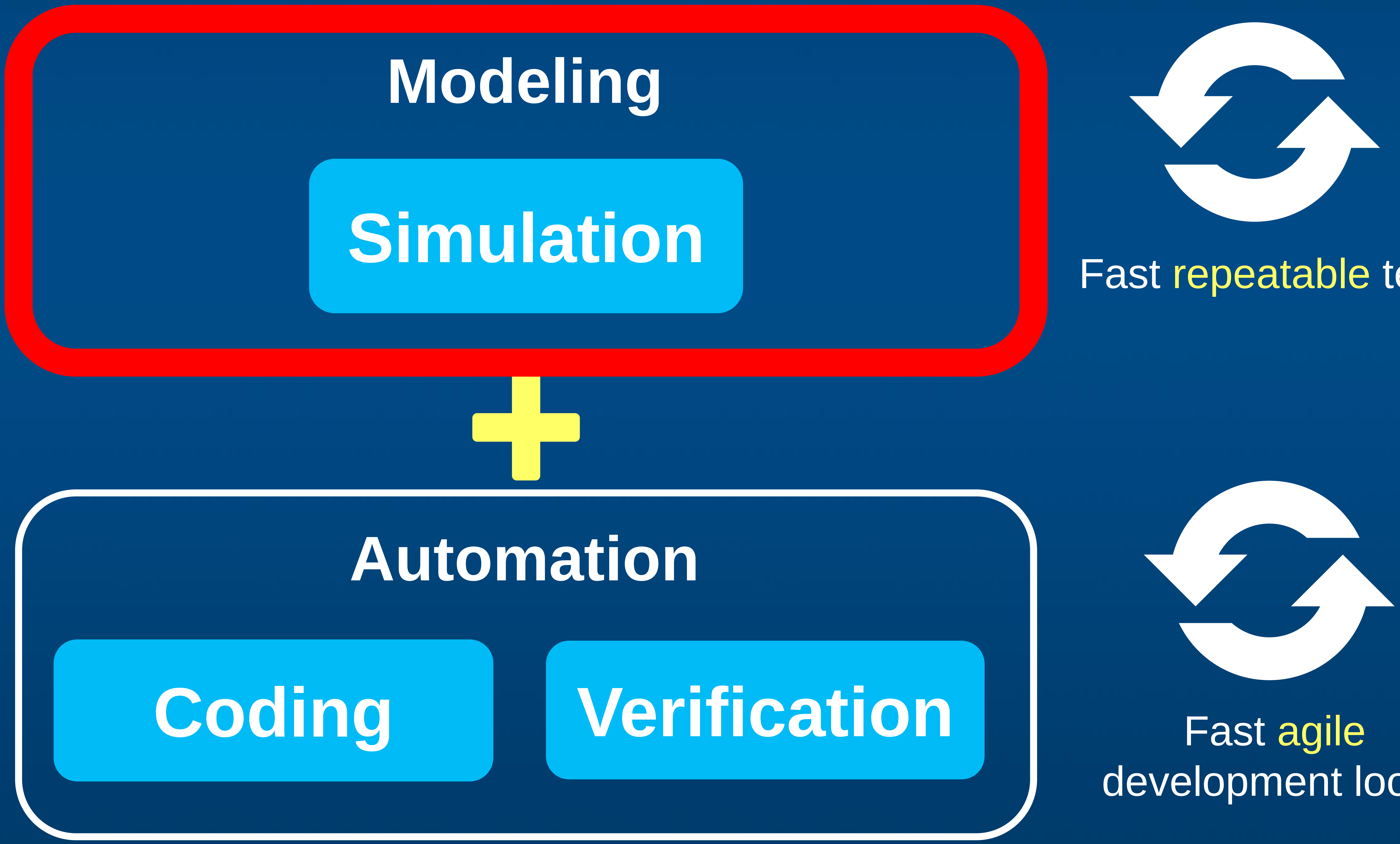


Simulation helps achieve this improbable task

*Source: Center for Artificial Intelligence, Saarland University

Model-Based Design

Systematic use of models throughout the development process



Modeling

Simulation

Automation

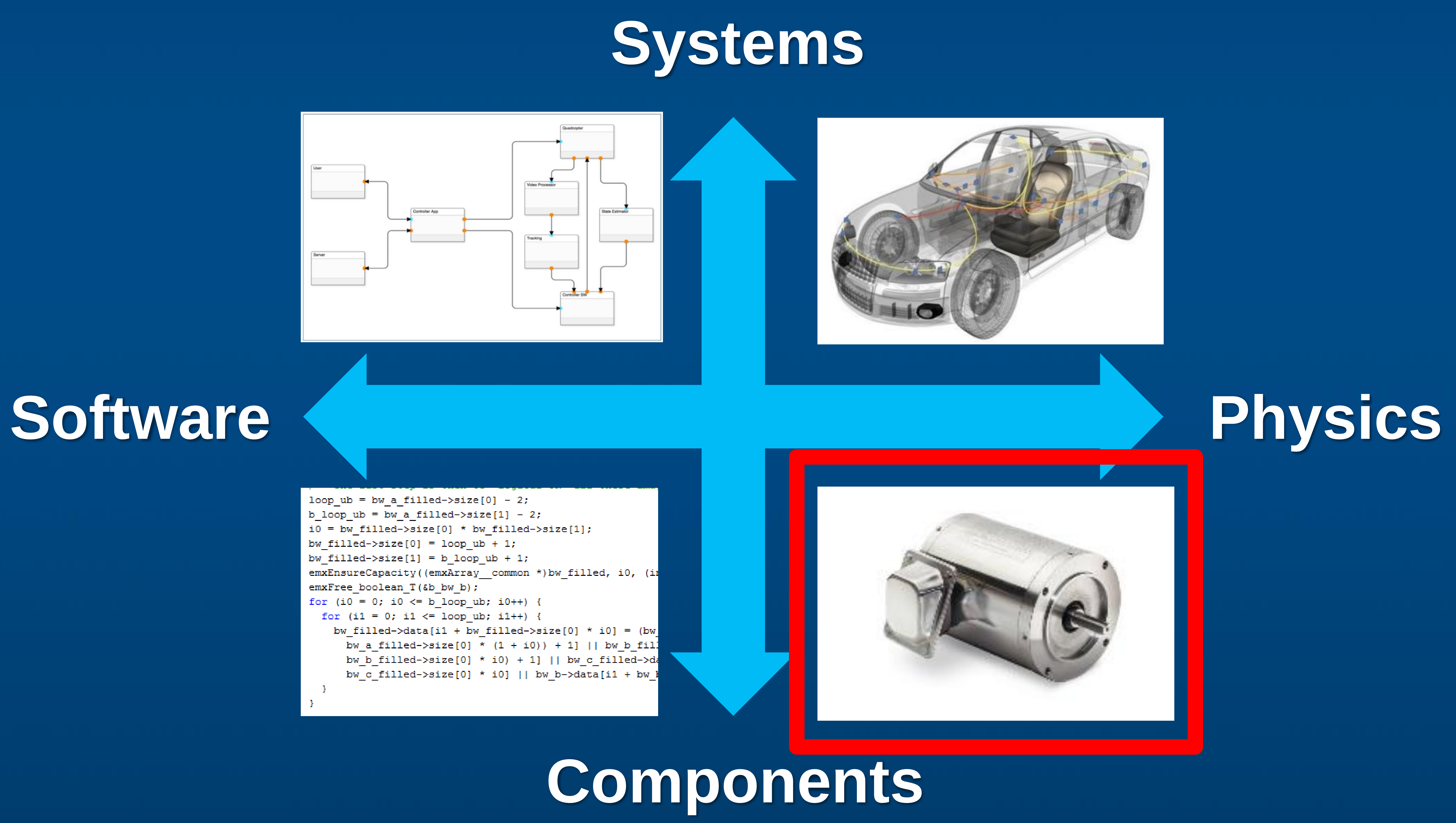
Coding

Verification

Fast repeatable tests

Fast agile development loops

Types of models



Systems

Software

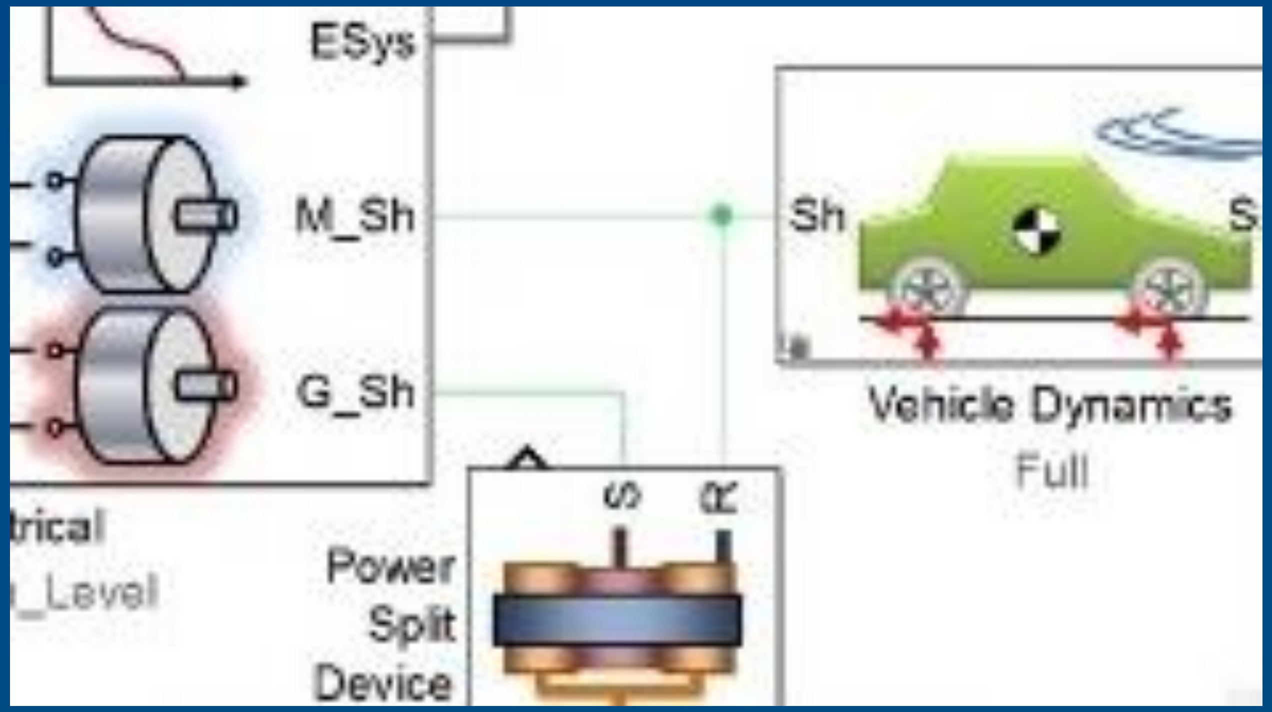
Physics

Components

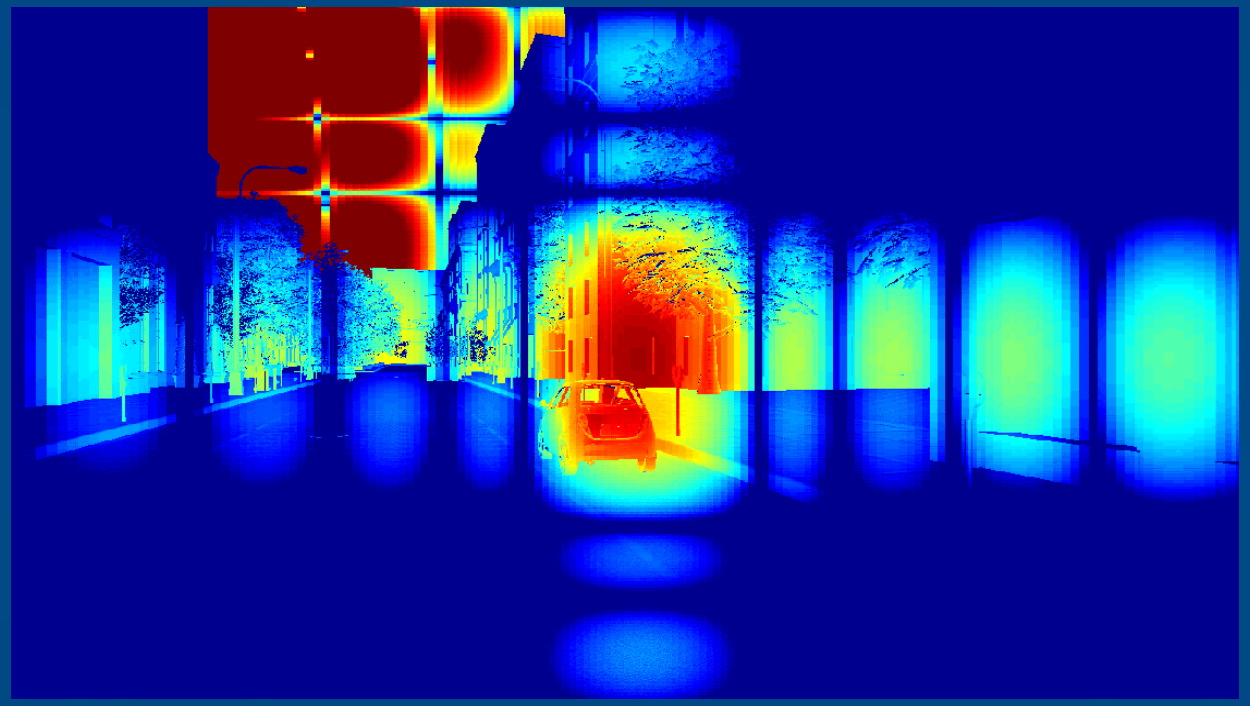
```
loop_ub = bw_a_filled->size[0] - 2;
b_loop_ub = bw_a_filled->size[1] - 2;
i0 = bw_filled->size[0] * bw_filled->size[1];
bw_filled->size[0] = loop_ub + 1;
bw_filled->size[1] = b_loop_ub + 1;
emxEnsureCapacity((emxArray__common *)bw_filled, i0, (i
emxFree_boolean_T(&b_bw_b);
for (i0 = 0; i0 <= b_loop_ub; i0++) {
  for (i1 = 0; i1 <= loop_ub; i1++) {
    bw_filled->data[i1 + bw_filled->size[0] * i0] = (bw_
      bw_a_filled->size[0] * (1 + i0)) + 1] || bw_b_fil
      bw_b_filled->size[0] * i0) + 1] || bw_c_filled->da
      bw_c_filled->size[0] * i0] || bw_b->data[i1 + bw_
  }
}
```

Physical components

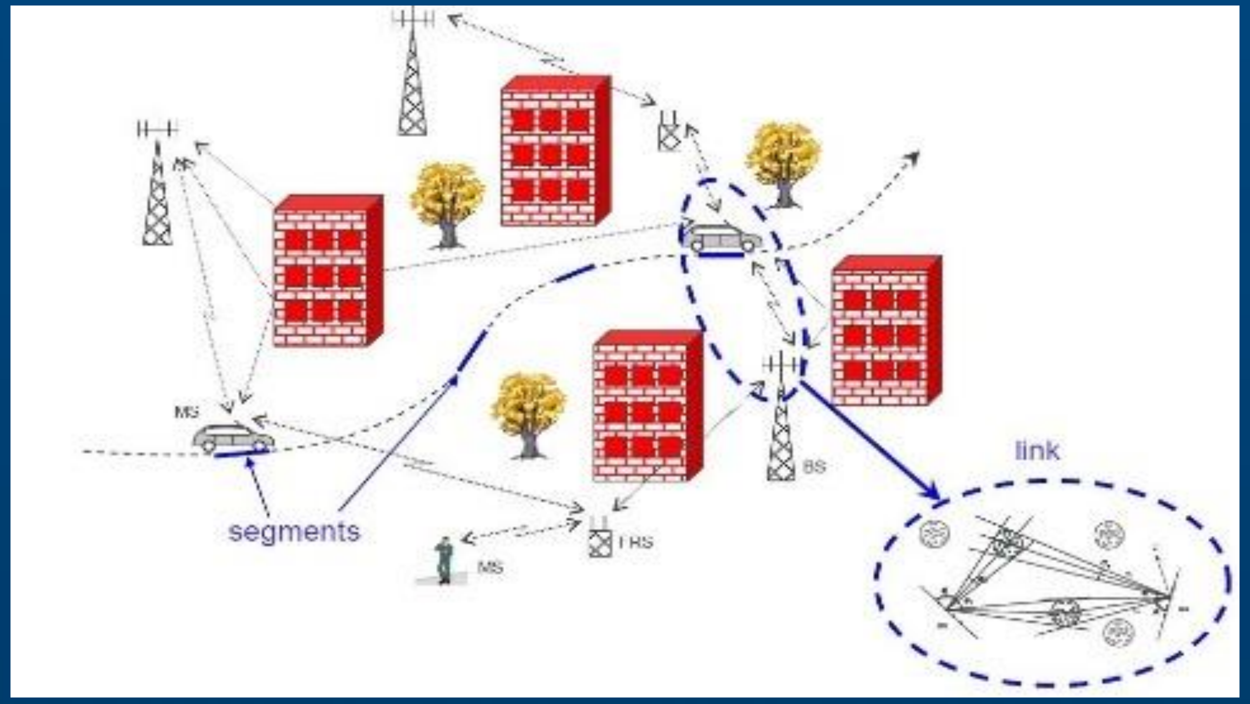
Vehicle Component



Sensor Model

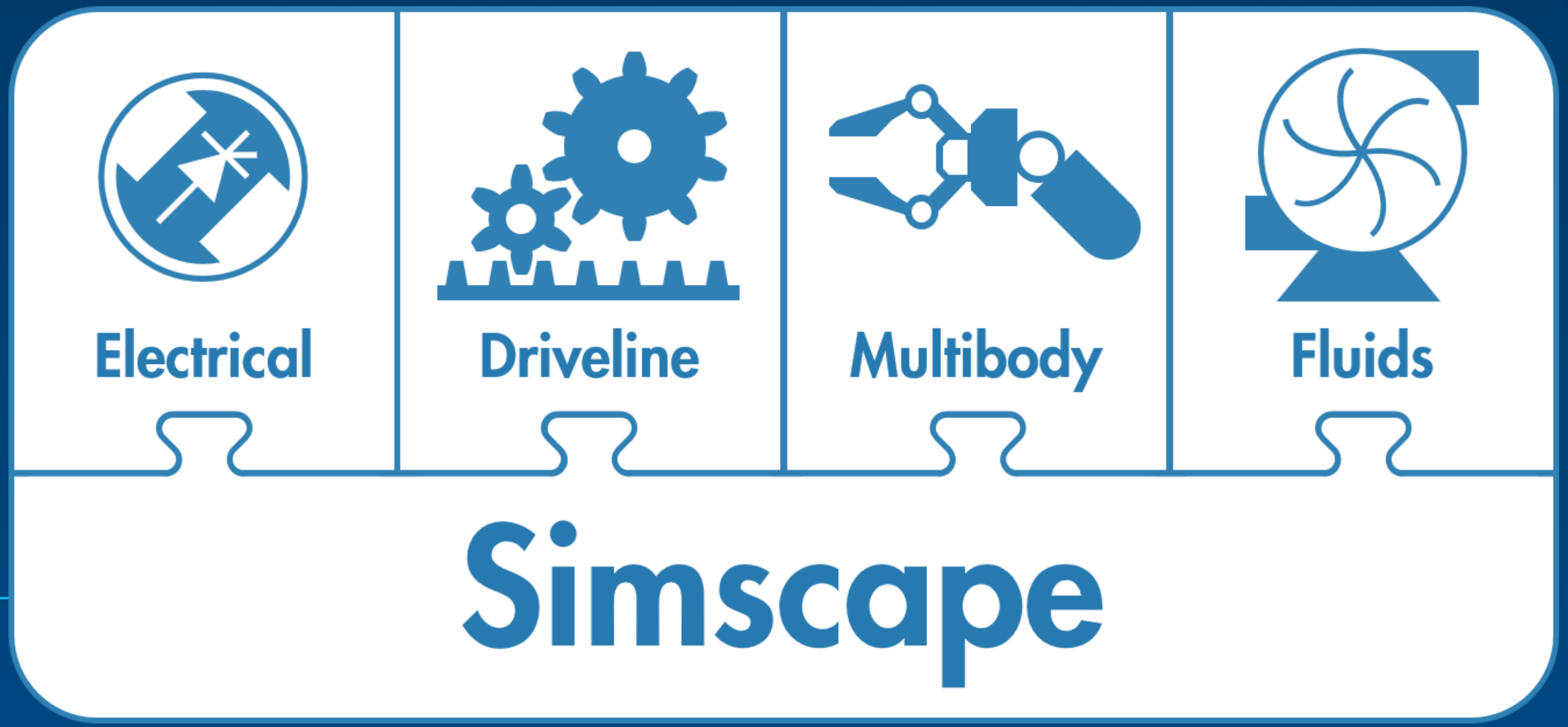


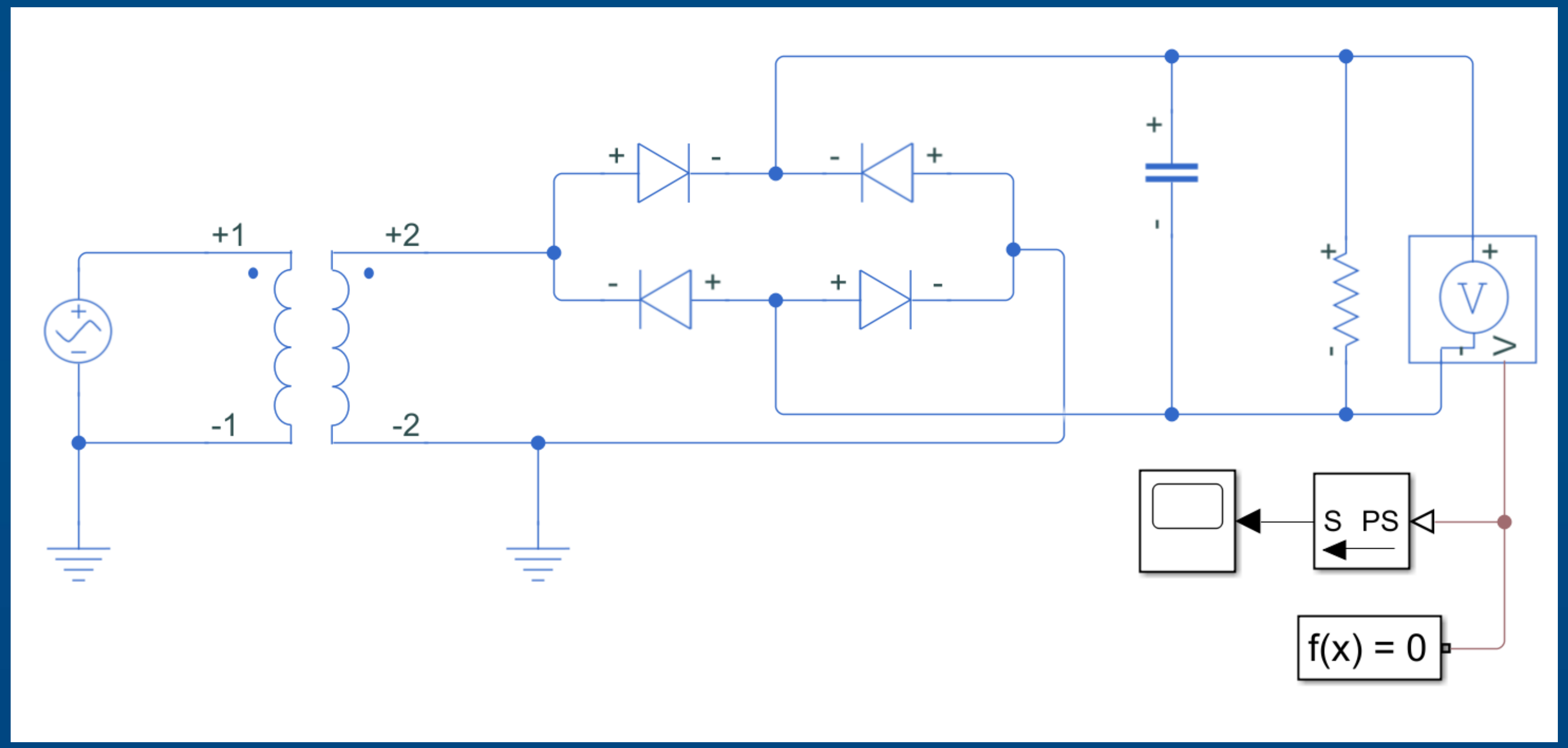
Communications Channel

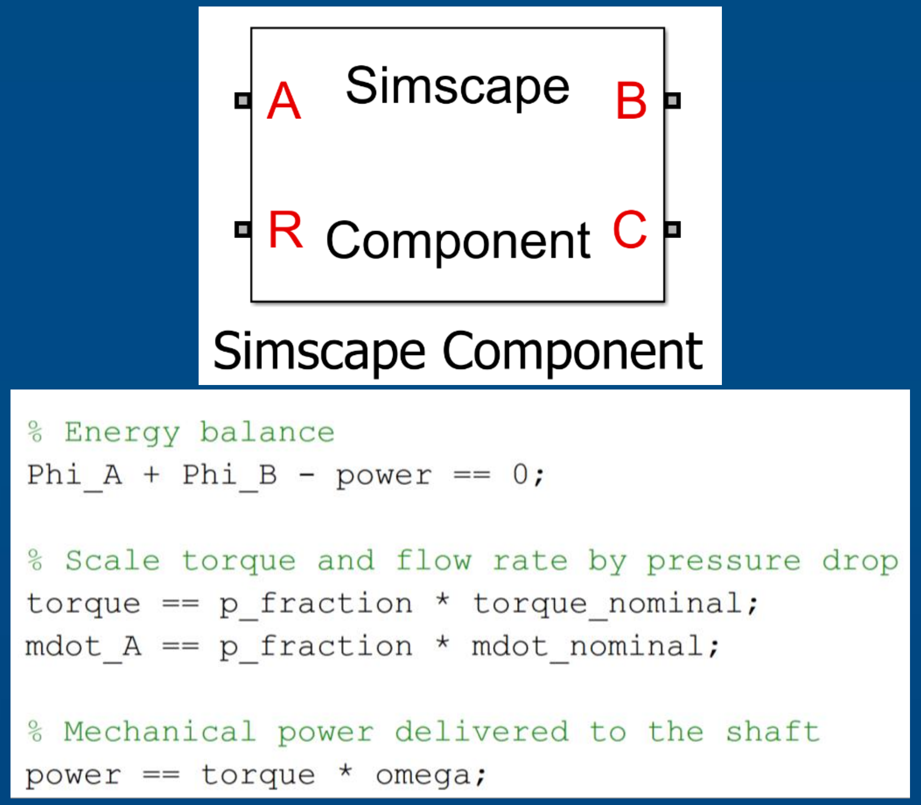


Motor

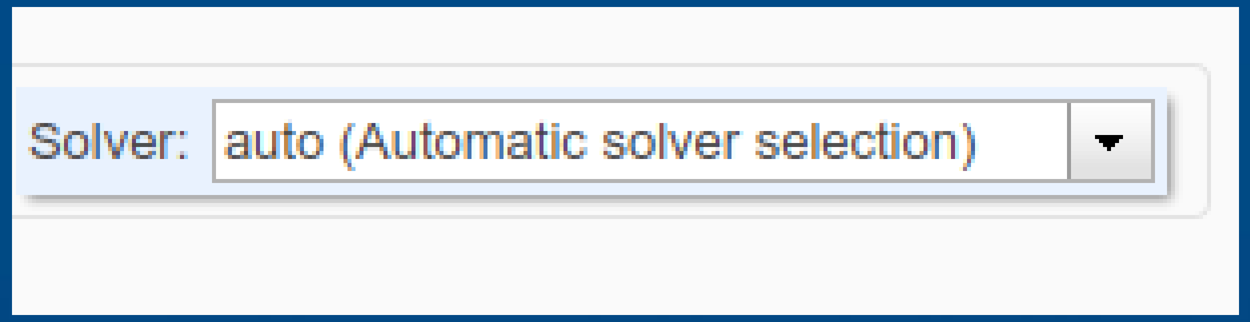
Simscape for physical modeling







```
% Energy balance
Phi_A + Phi_B - power == 0;

% Scale torque and flow rate by pressure drop
torque == p_fraction * torque_nominal;
mdot_A == p_fraction * mdot_nominal;

% Mechanical power delivered to the shaft
power == torque * omega;
```



Solver: auto (Automatic solver selection)

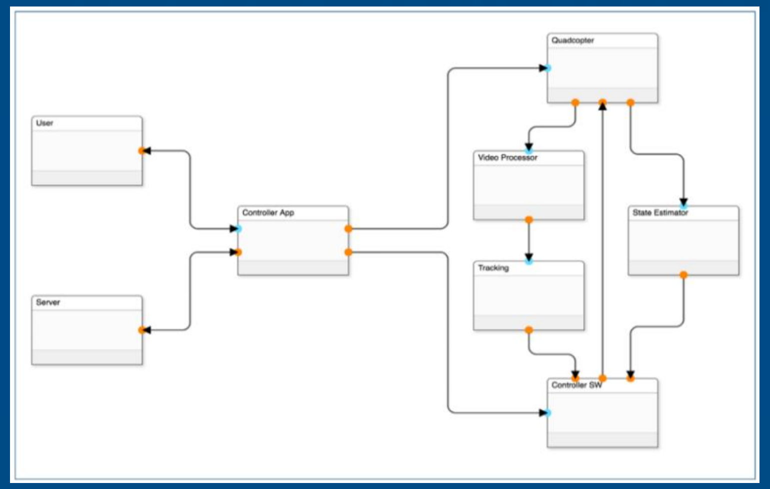
Publication-quality diagrams

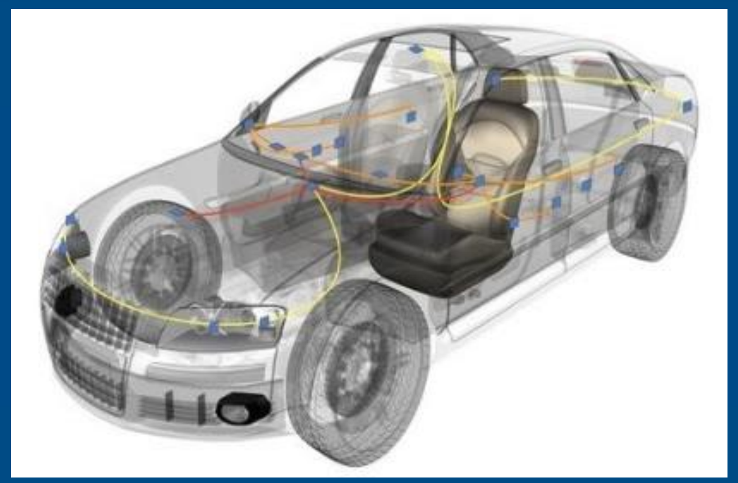
Simscape modeling language

Models just run

Types of models

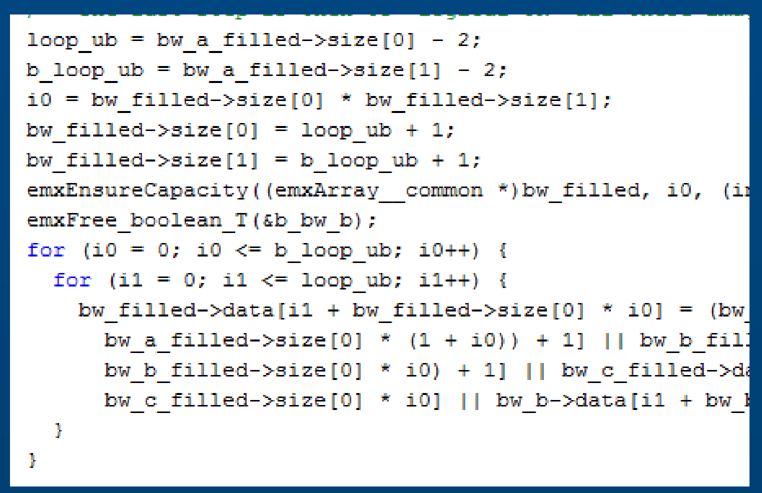
Systems





Software

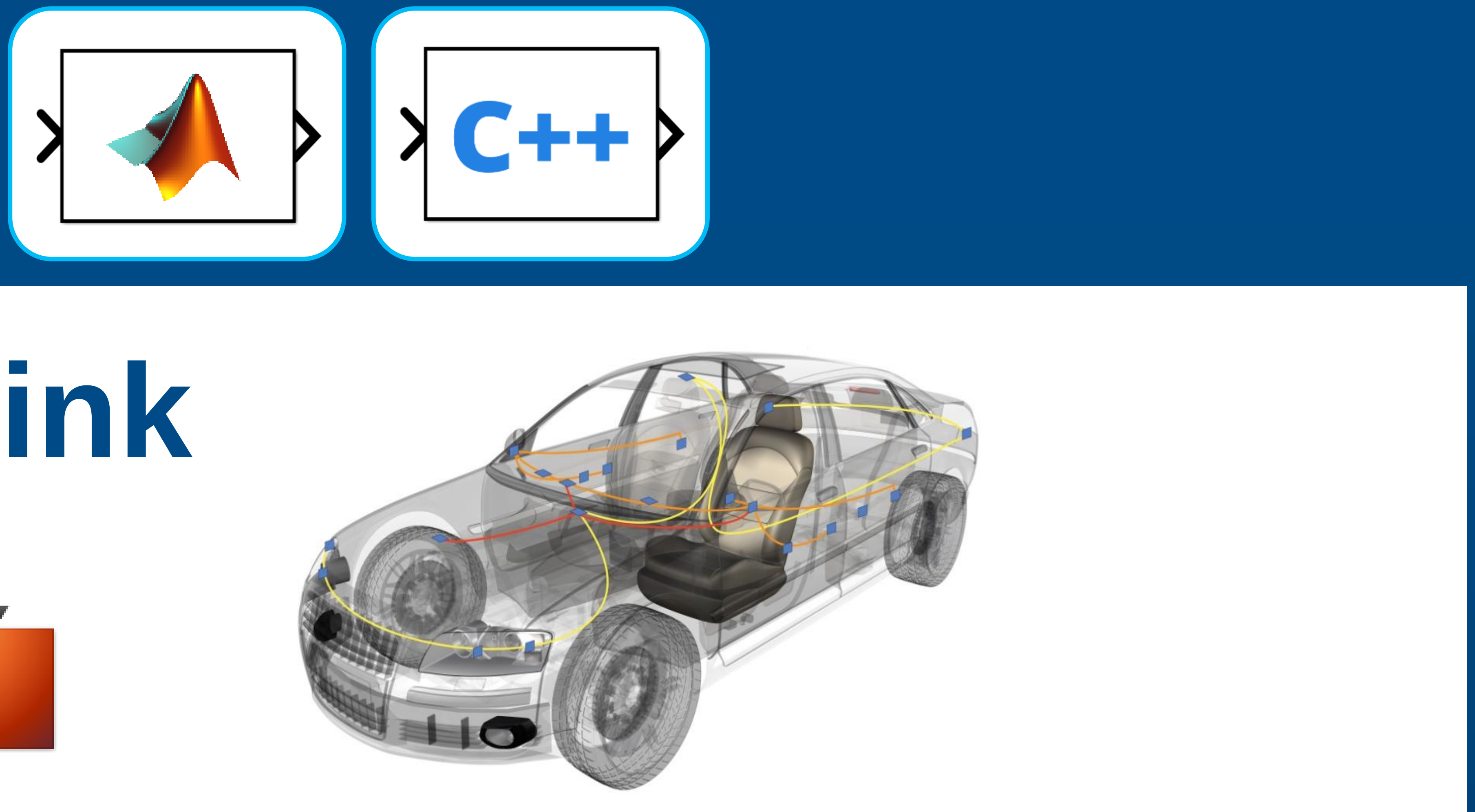
Physics



```
loop_ub = bw_a_filled->size[0] - 2;
b_loop_ub = bw_a_filled->size[1] - 2;
i0 = bw_filled->size[0] * bw_filled->size[1];
bw_filled->size[0] = loop_ub + 1;
bw_filled->size[1] = b_loop_ub + 1;
emxEnsureCapacity((emxArray__common *)bw_filled, i0, (i
emxFree_boolean_T(&b_bw_b);
for (i0 = 0; i0 <= b_loop_ub; i0++) {
  for (i1 = 0; i1 <= loop_ub; i1++) {
    bw_filled->data[i1 + bw_filled->size[0] * i0] = (bw_
      bw_a_filled->size[0] * (1 + i0)) + 1] || bw_b_fil
      bw_b_filled->size[0] * i0) + 1] || bw_c_filled->da
      bw_c_filled->size[0] * i0] || bw_b->data[i1 + bw_
  }
}
```

Components

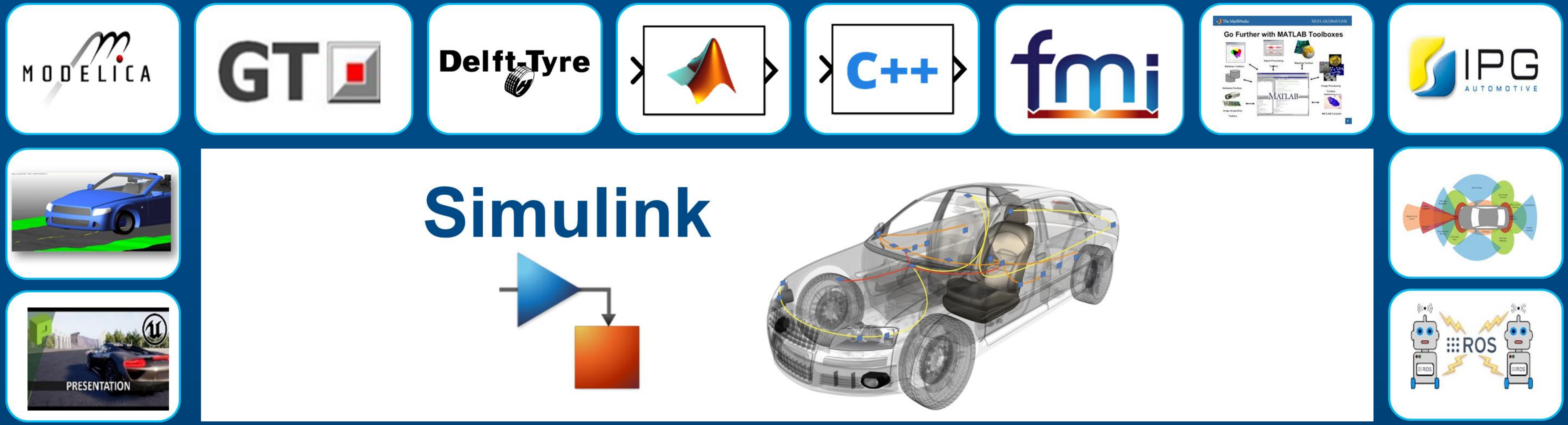
Simulink as an Integration Platform



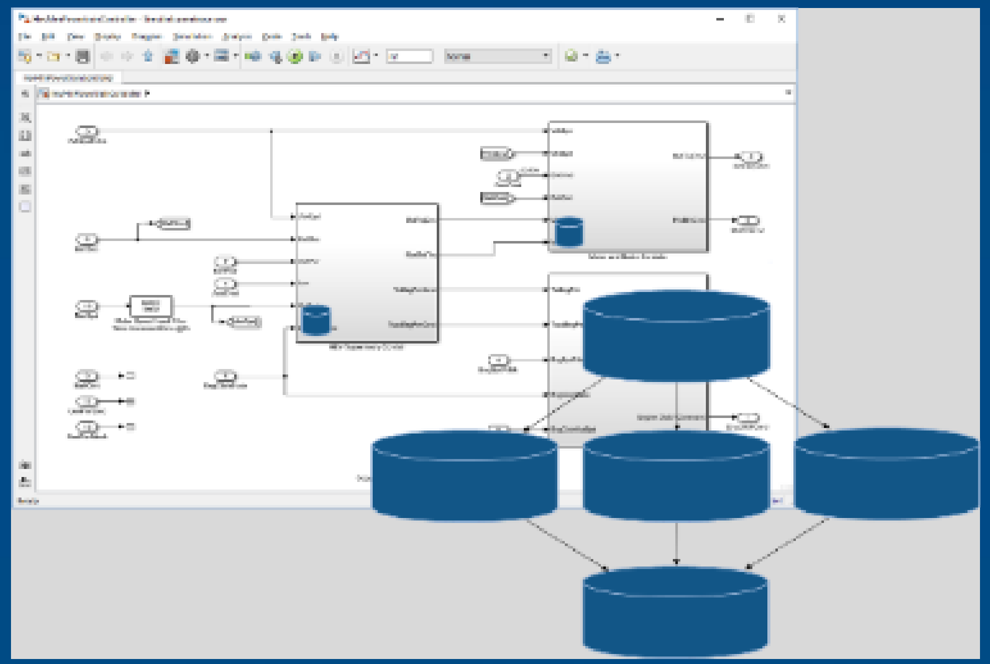
C++

Simulink

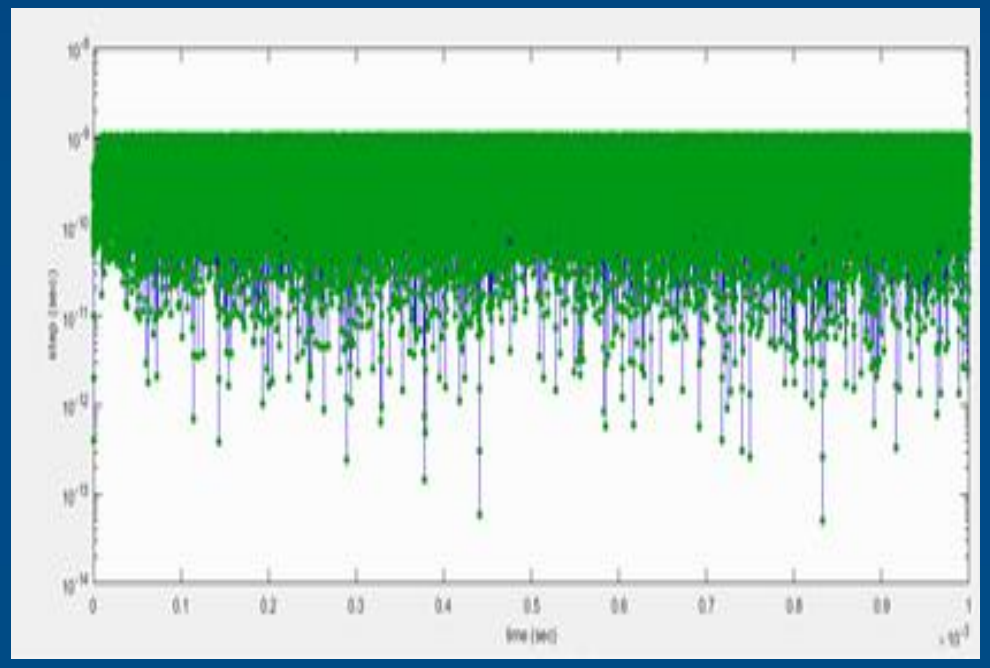
Simulink as an Integration Platform



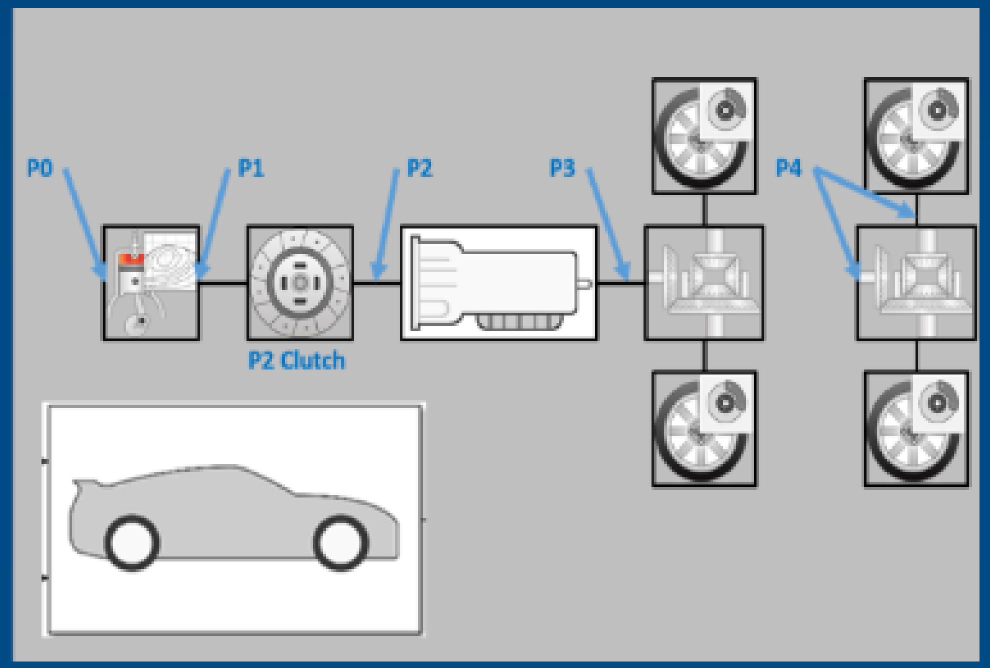
Simulation Integration: Infrastructure



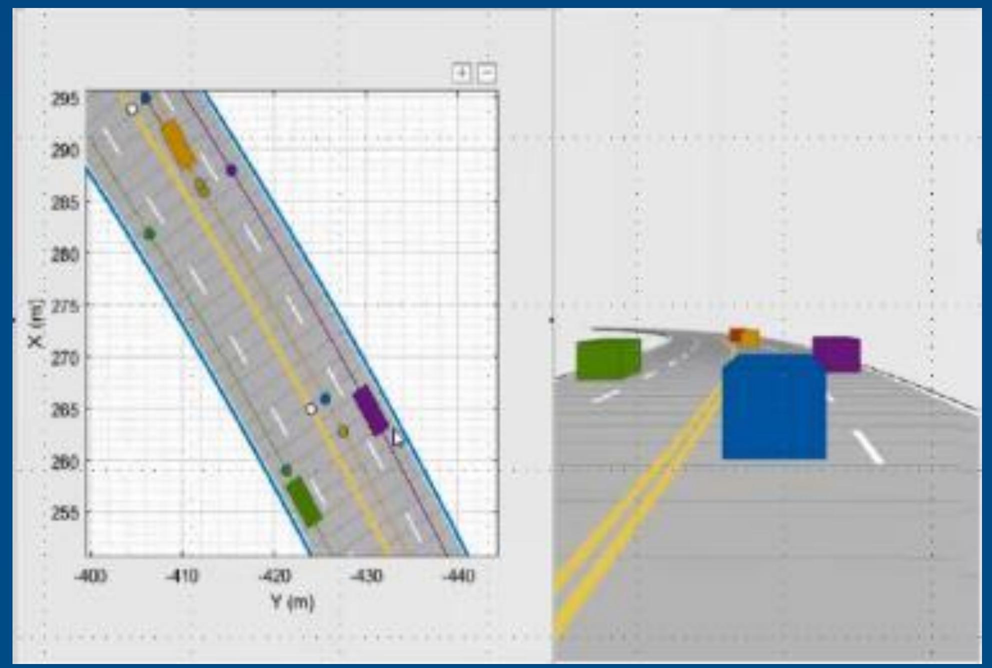
Data Management



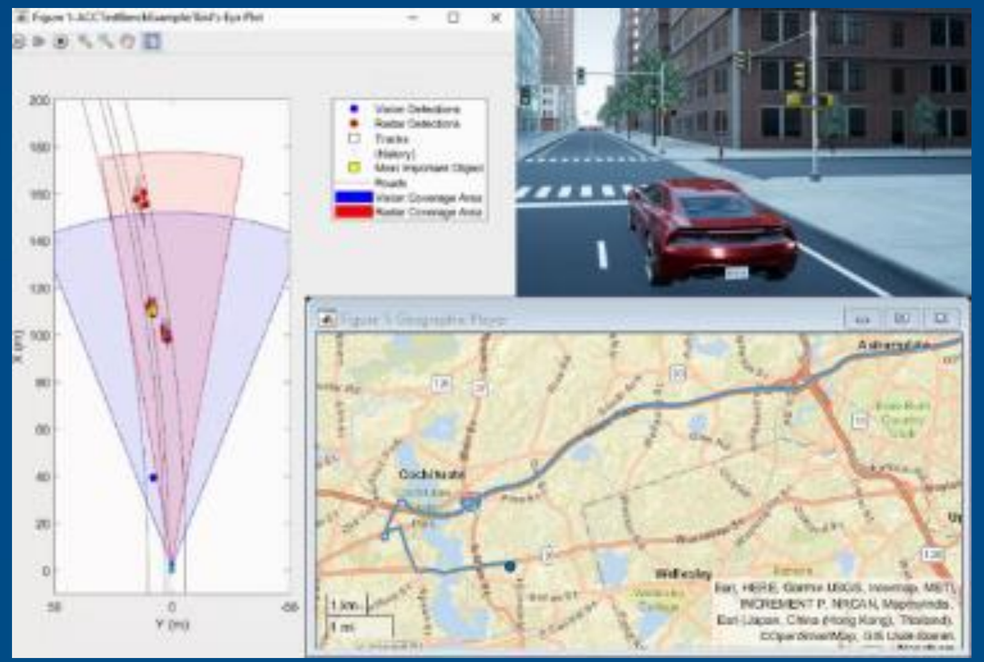
Solver Technology



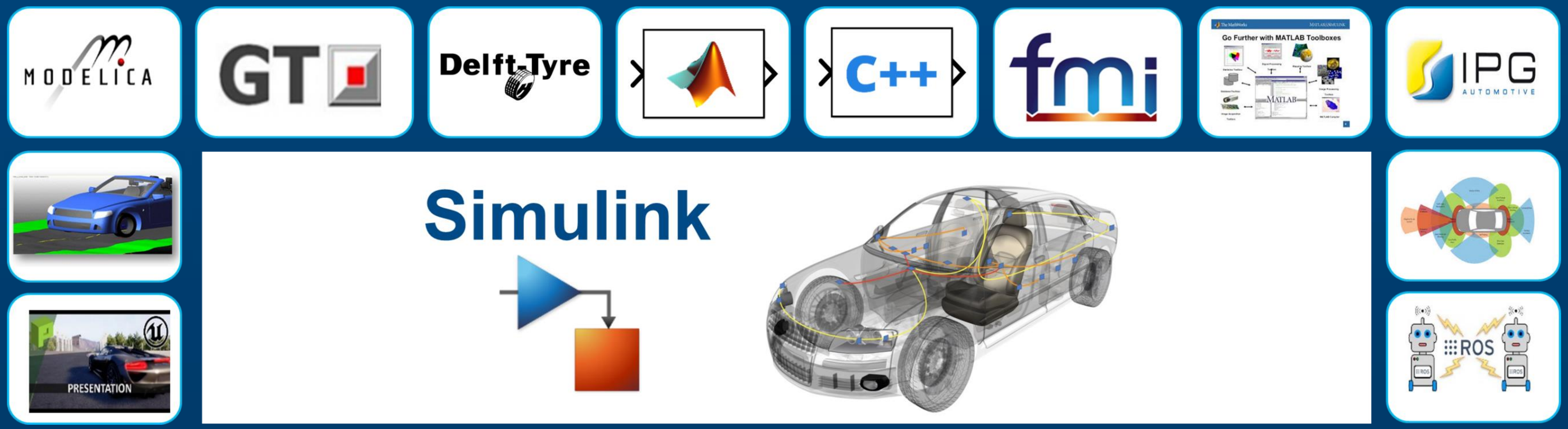
Vehicle Configuration



Multi-actor Scenarios



Visualization



Simulation Integration: Analyses

Verification and Validation

Design Optimization

Sensitivity Analysis

Virtual Calibration

Pure EV
(will update graphics)

Vehicle Dynamics
(will update graphics)

Hybrid EV
(will update graphics)

Automated Driving
(will update graphics)

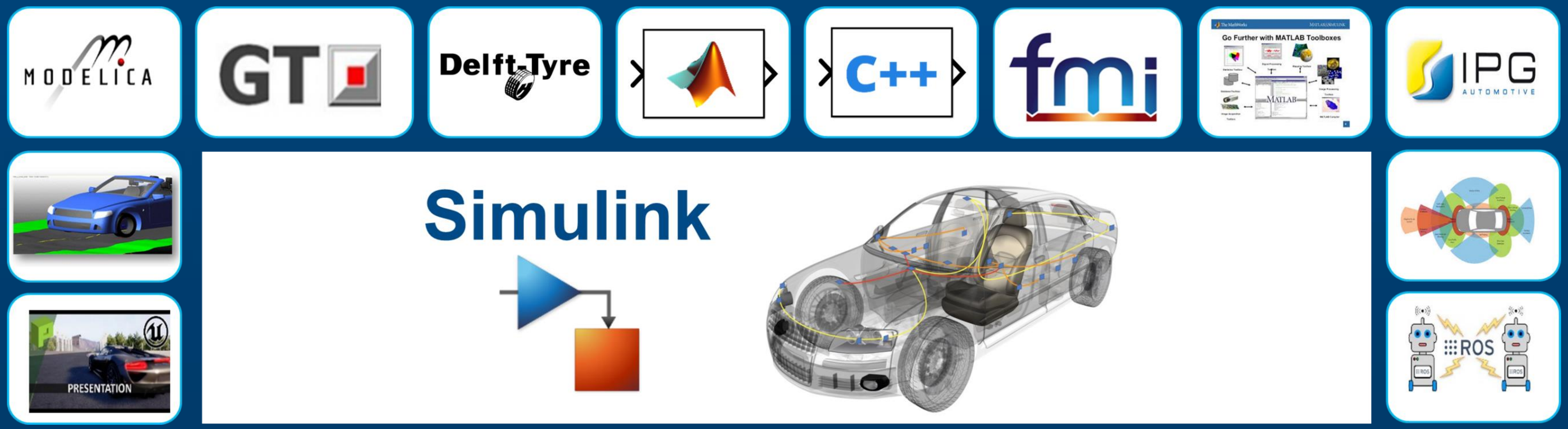
Fuel Economy

Performance

Energy Consumption

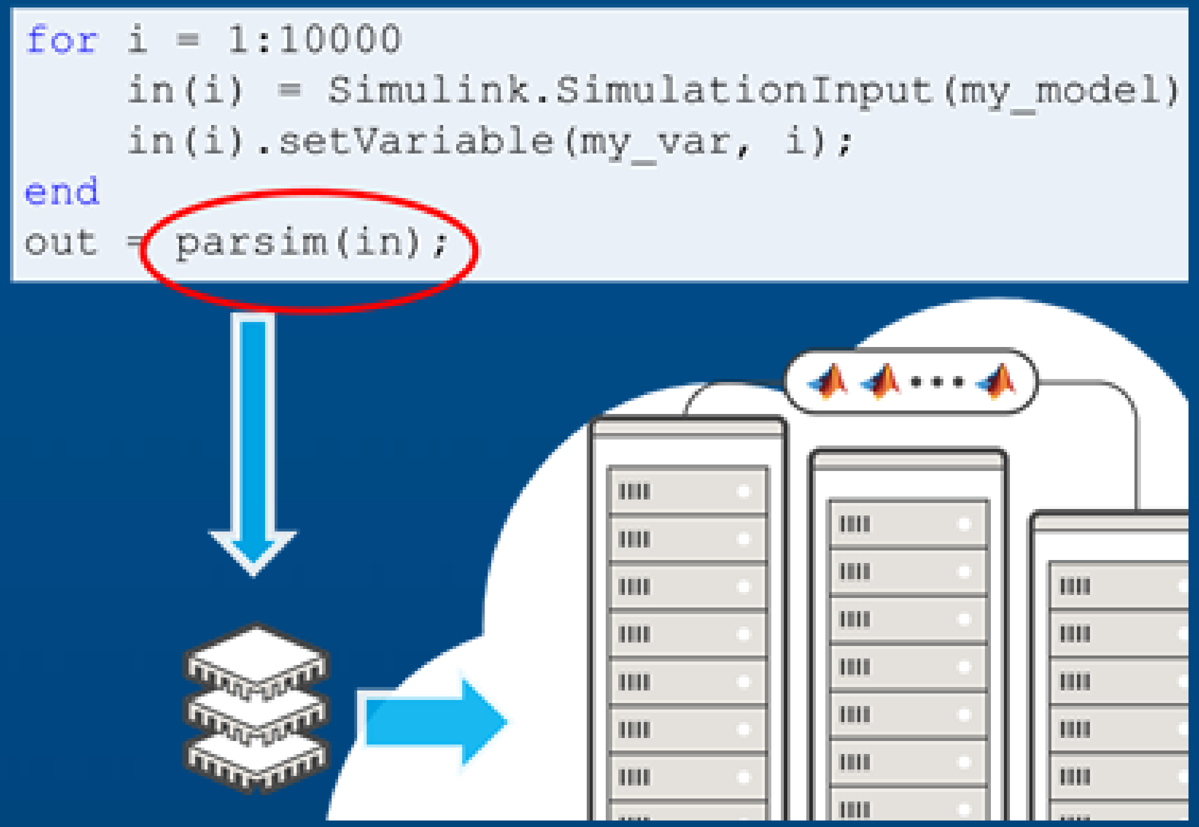
Drivability

Ride & Handling



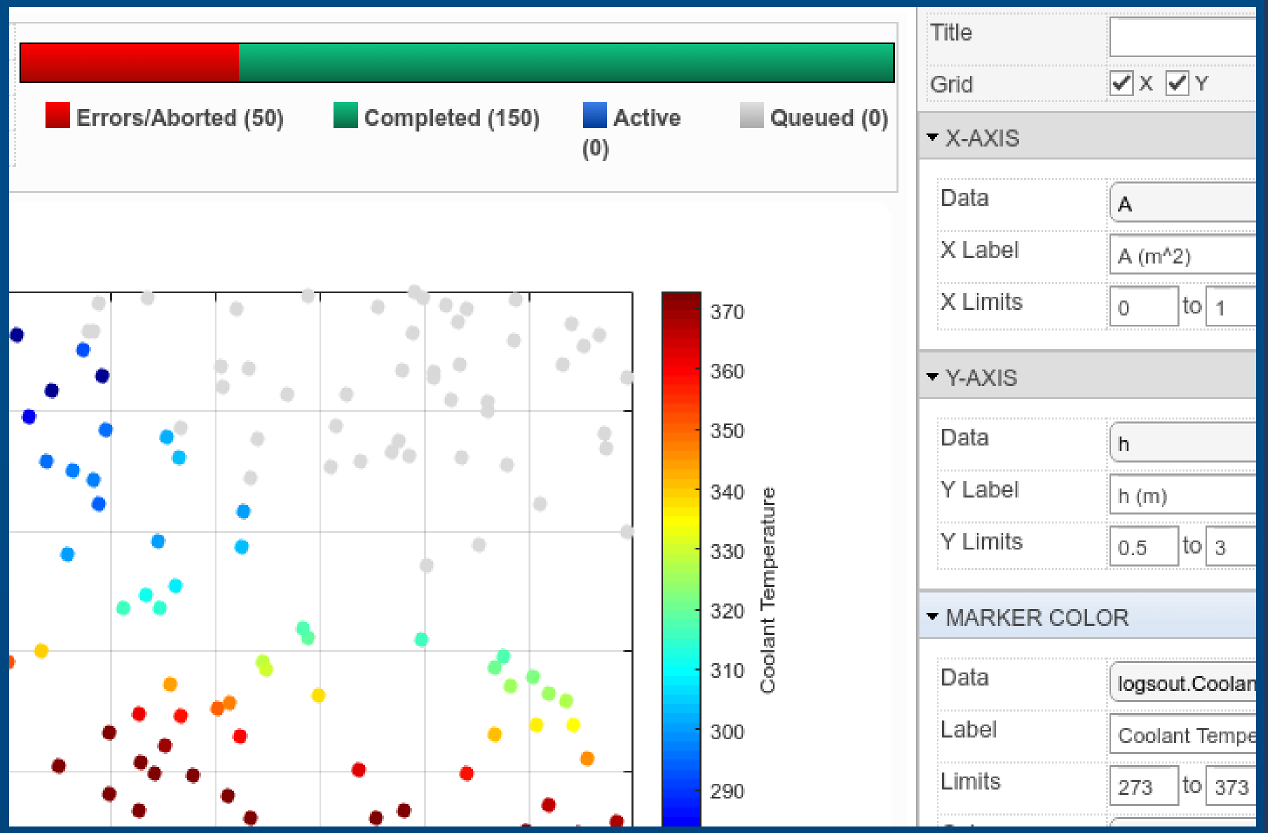
Scaling up simulations

X 1,000,000’s



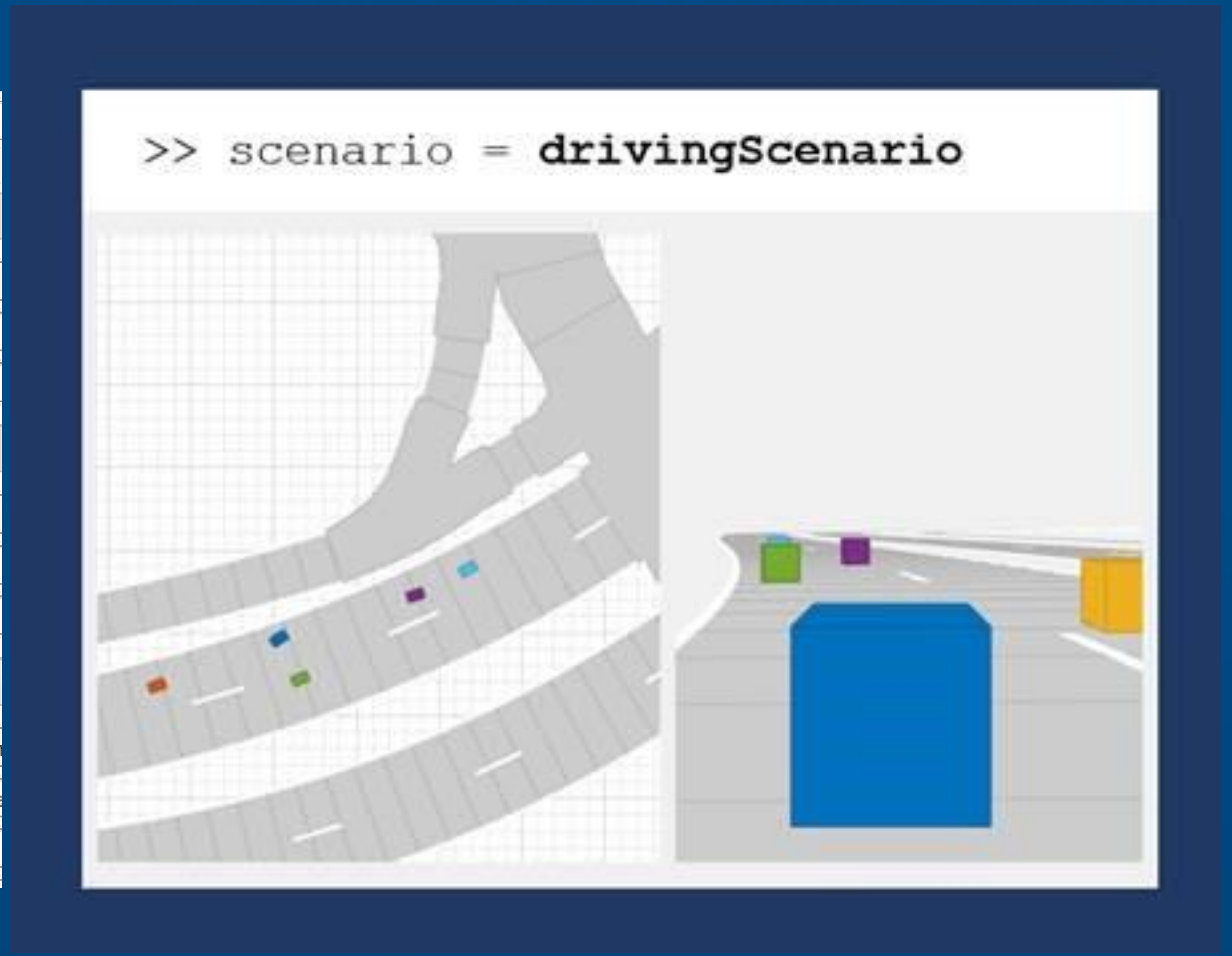
```
for i = 1:10000
    in(i) = Simulink.SimulationInput(my_model)
    in(i).setVariable(my_var, i);
end
out = parsim(in);
```

Parallel simulations



Title
Grid ☑ X ☑ Y
X-AXIS
Data A
X Label A (m^2)
X Limits 0 to 1
Y-AXIS
Data h
Y Label h (m)
Y Limits 0.5 to 3
MARKER COLOR
Data logsout.Coolan
Label Coolant Tempe
Limits 273 to 373

Simulation Manager

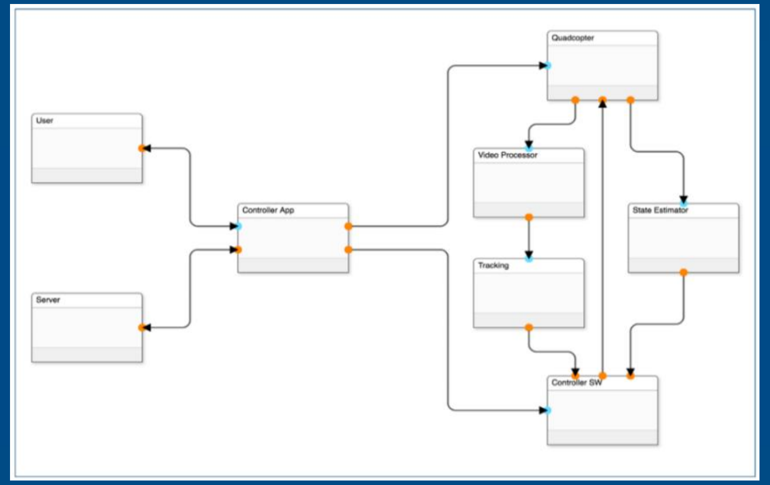


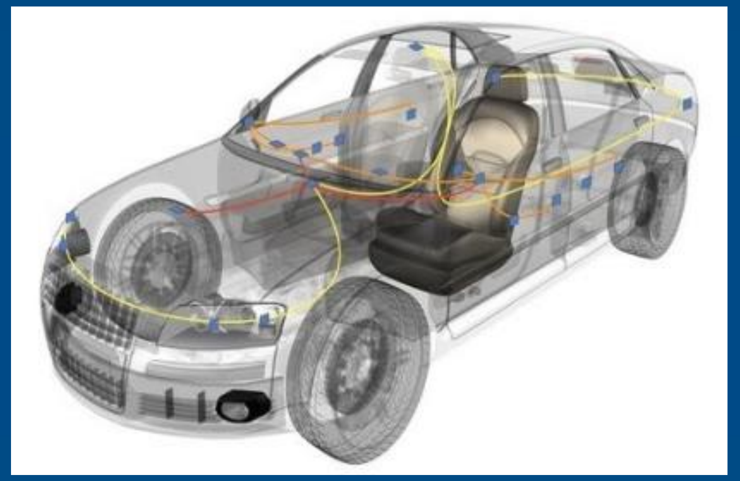
```
>> scenario = drivingScenario
```

Programmatic test creation

Types of models

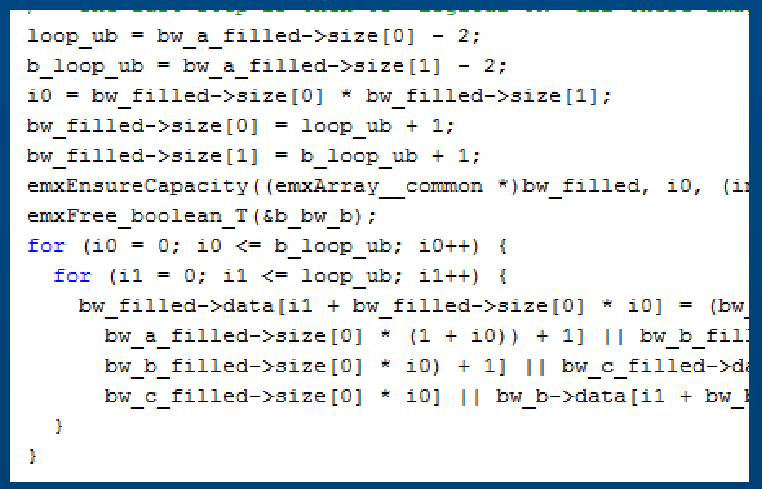
Systems





Software

Physics

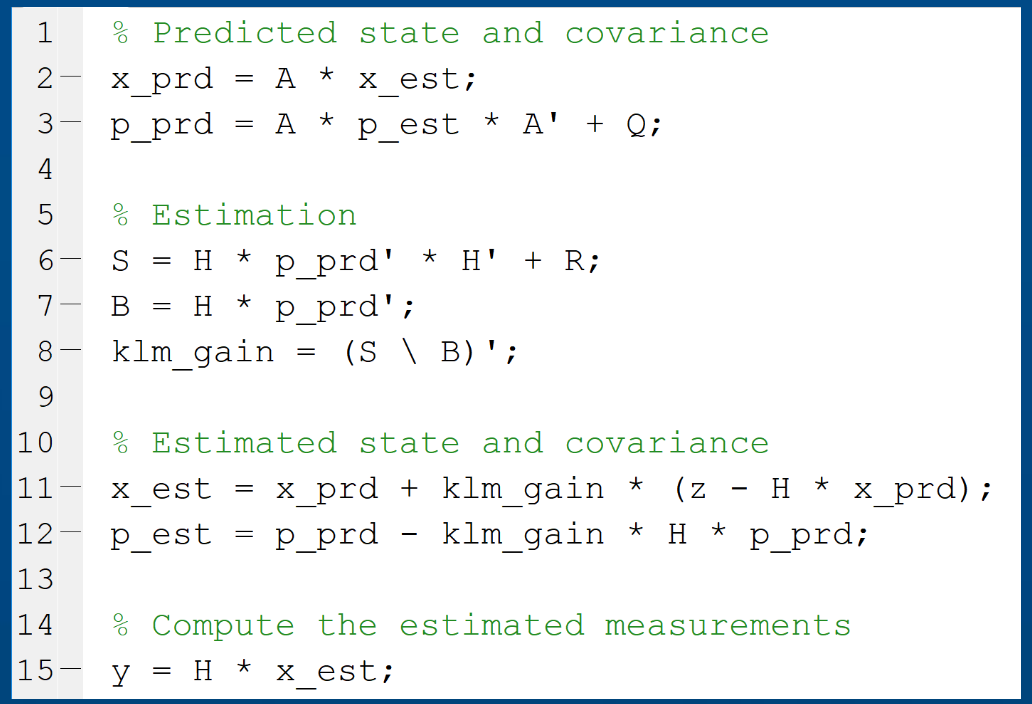


```
loop_ub = bw_a_filled->size[0] - 2;
b_loop_ub = bw_a_filled->size[1] - 2;
i0 = bw_filled->size[0] * bw_filled->size[1];
bw_filled->size[0] = loop_ub + 1;
bw_filled->size[1] = b_loop_ub + 1;
emxEnsureCapacity((emxArray__common *)bw_filled, i0, (i
emxFree_boolean_T(&b_bw_b);
for (i0 = 0; i0 <= b_loop_ub; i0++) {
  for (i1 = 0; i1 <= loop_ub; i1++) {
    bw_filled->data[i1 + bw_filled->size[0] * i0] = (bw_
      bw_a_filled->size[0] * (1 + i0)) + 1] || bw_b_fil
      bw_b_filled->size[0] * i0) + 1] || bw_c_filled->da
      bw_c_filled->size[0] * i0] || bw_b->data[i1 + bw_
  }
}
```

Components

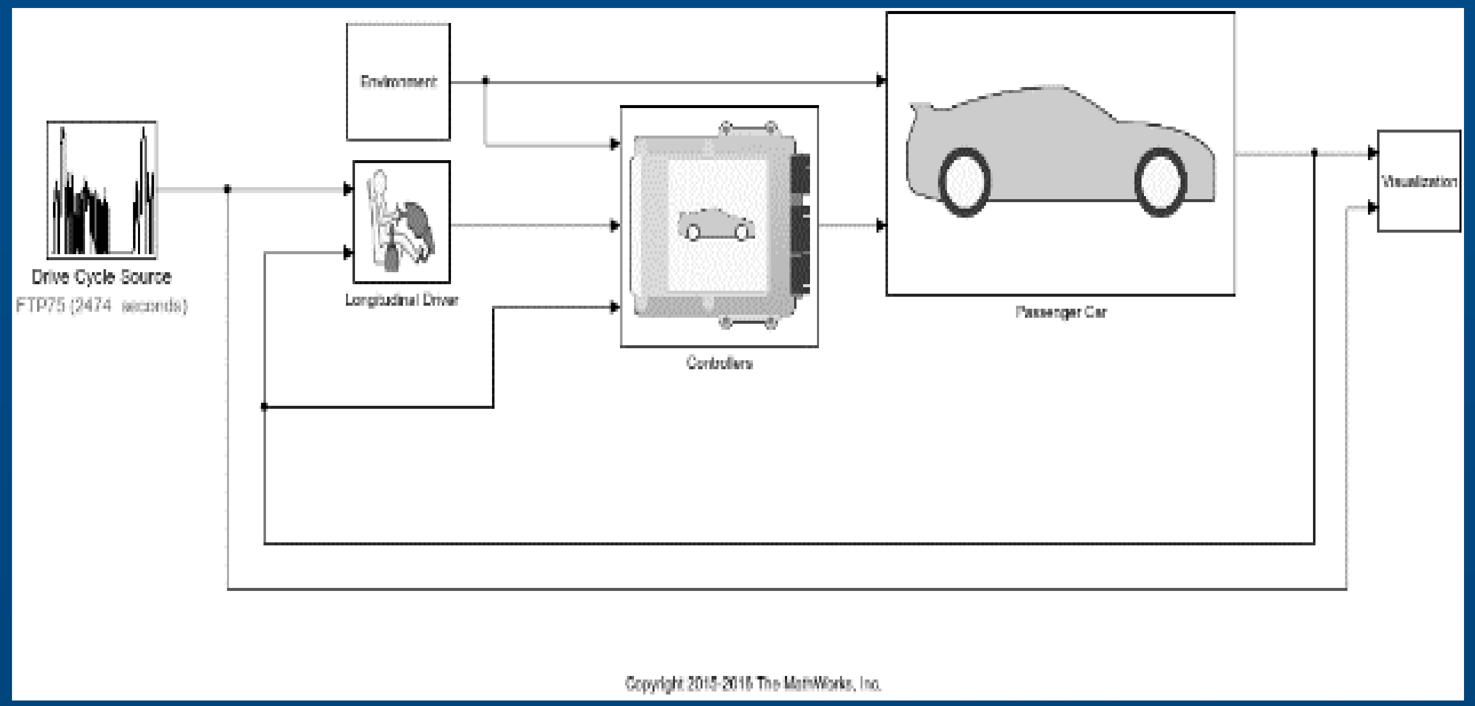
"A typical ECU contains 2000 function components that each are developed by a different person."

Working at a high-level of abstraction

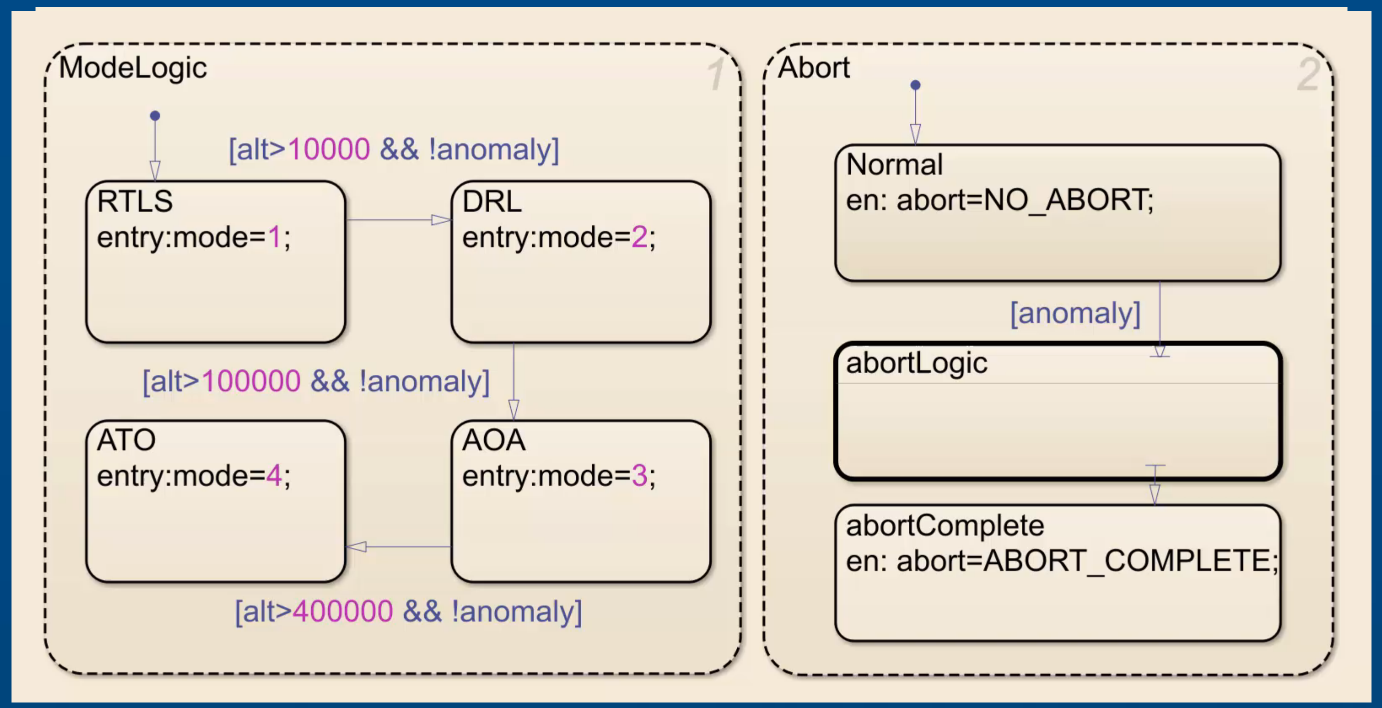


```
% Predicted state and covariance
x_prd = A * x_est;
p_prd = A * p_est * A' + Q;

% Estimation
S = H * p_prd' * H' + R;
B = H * p_prd';
klm_gain = (S \ B)';

% Estimated state and covariance
x_est = x_prd + klm_gain * (z - H * x_prd);
p_est = p_prd - klm_gain * H * p_prd;

% Compute the estimated measurements
y = H * x_est;
```

MATLAB

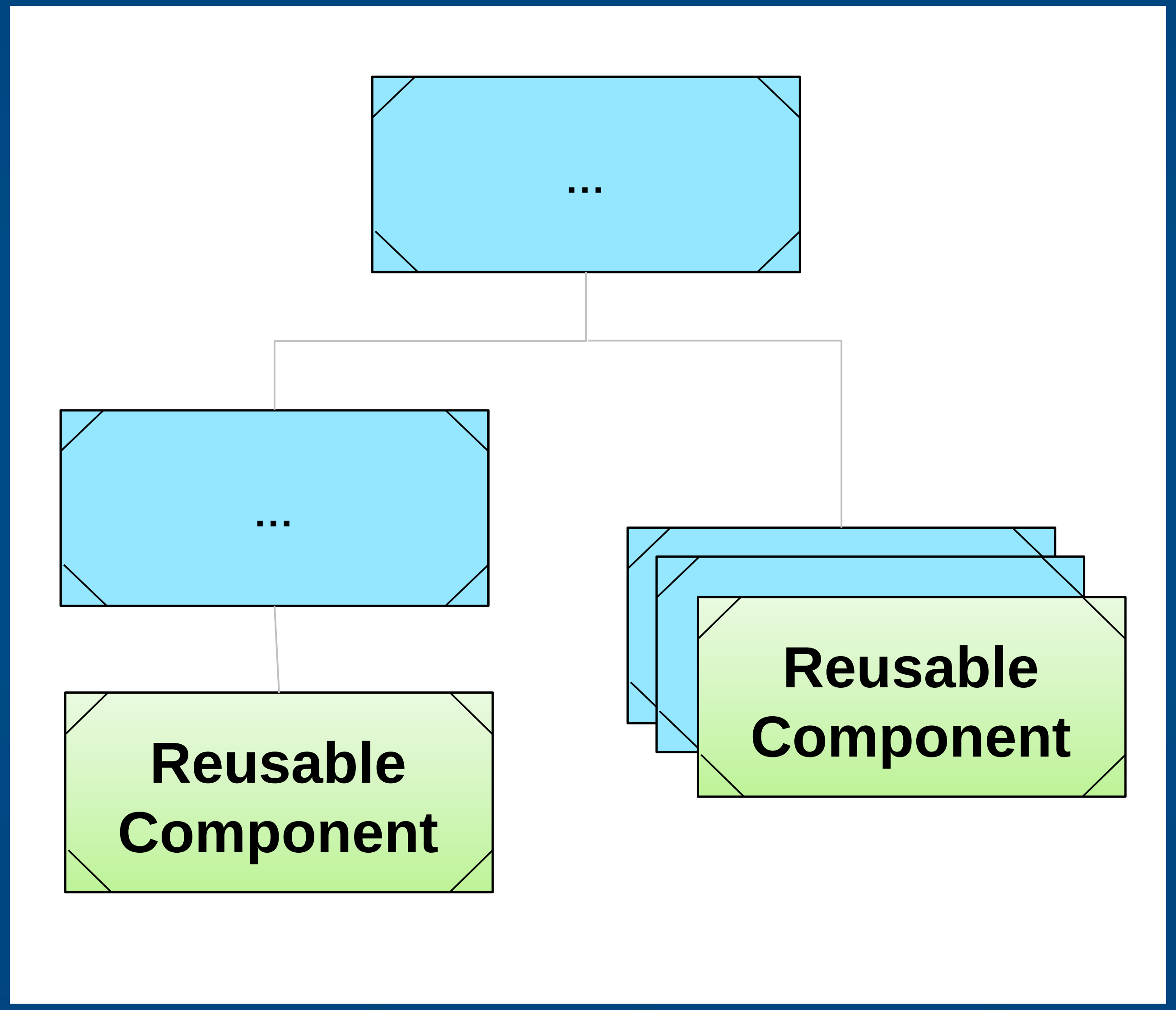


Simulink

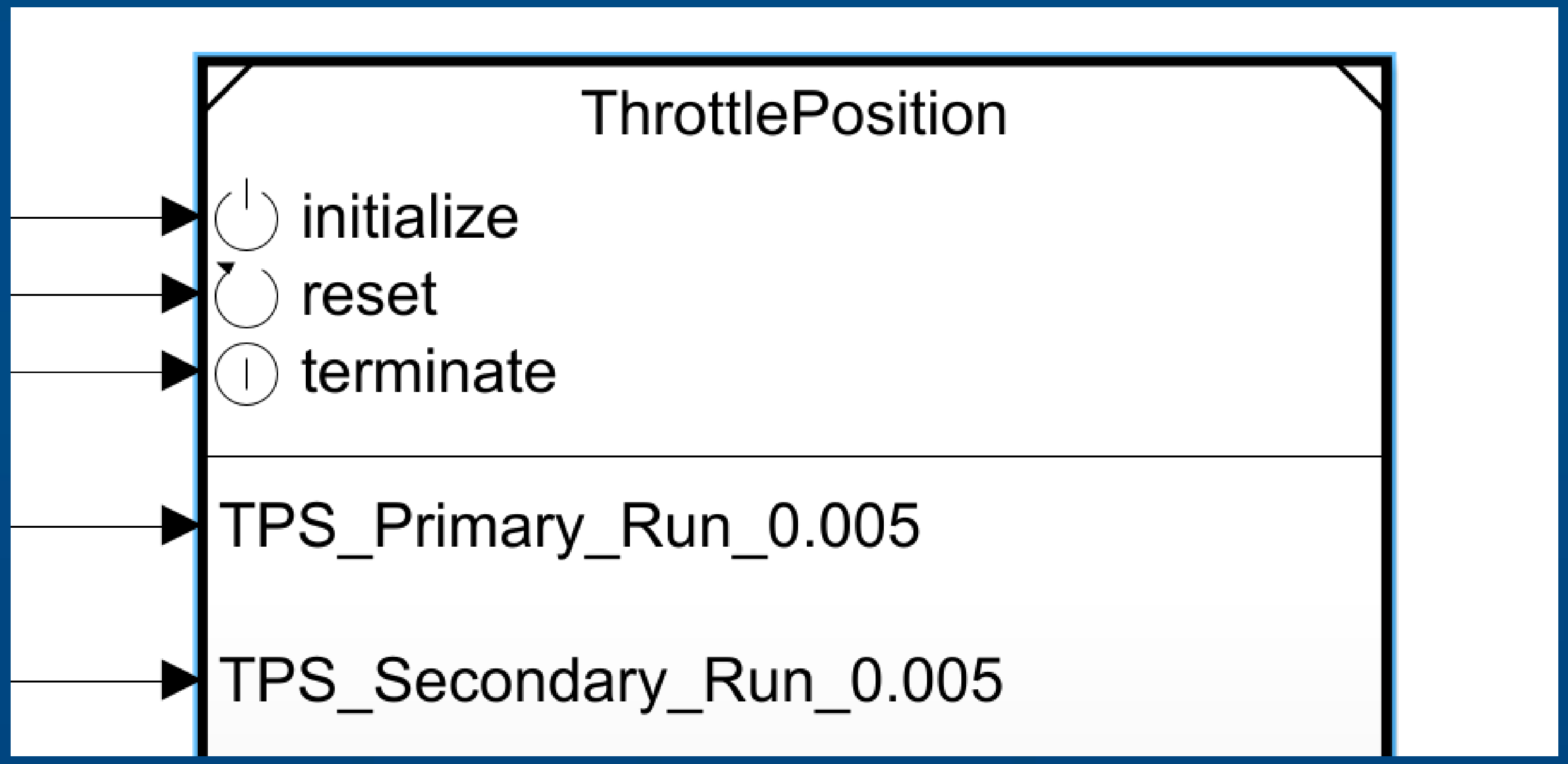


Stateflow

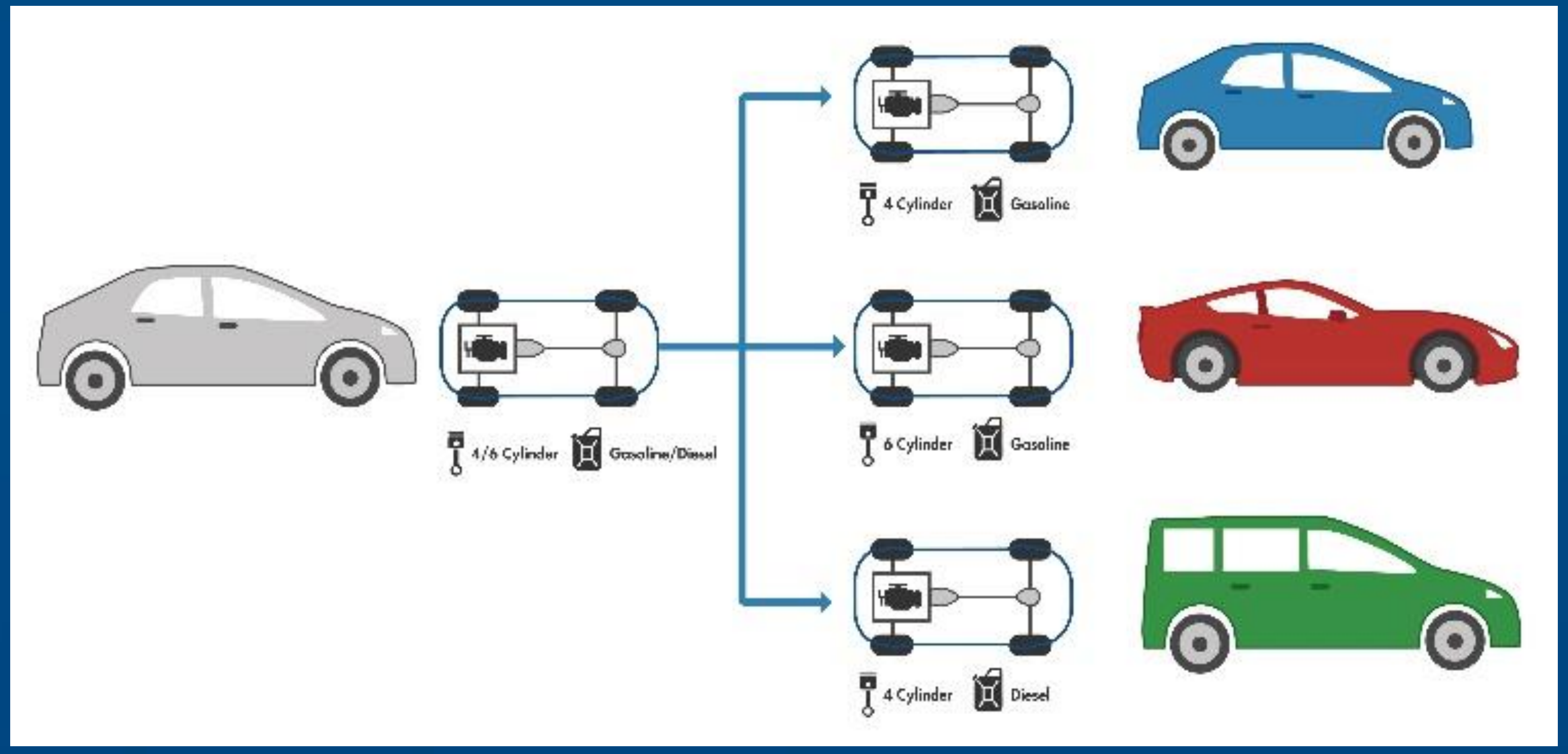
Component modeling



Reusable components that can be adapted to any software system



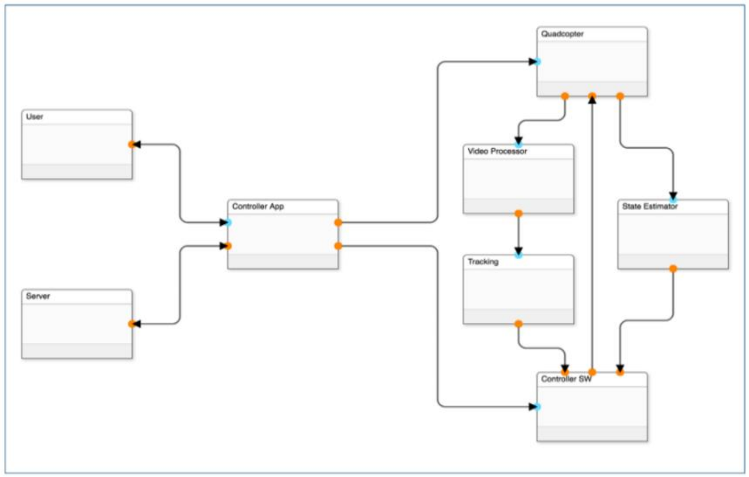
Startup and shutdown behavior

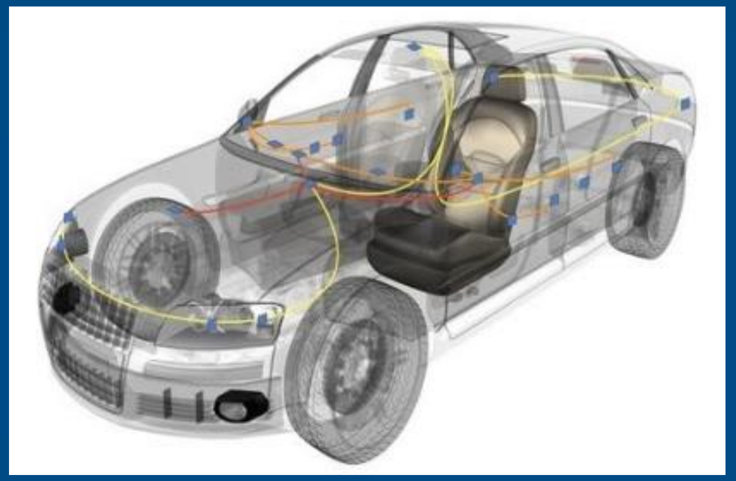


Variant management

Types of models

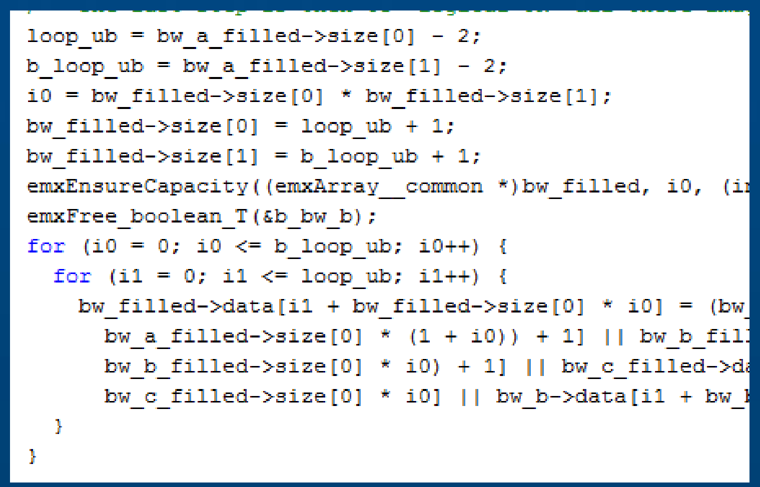
Systems





Software

Physics

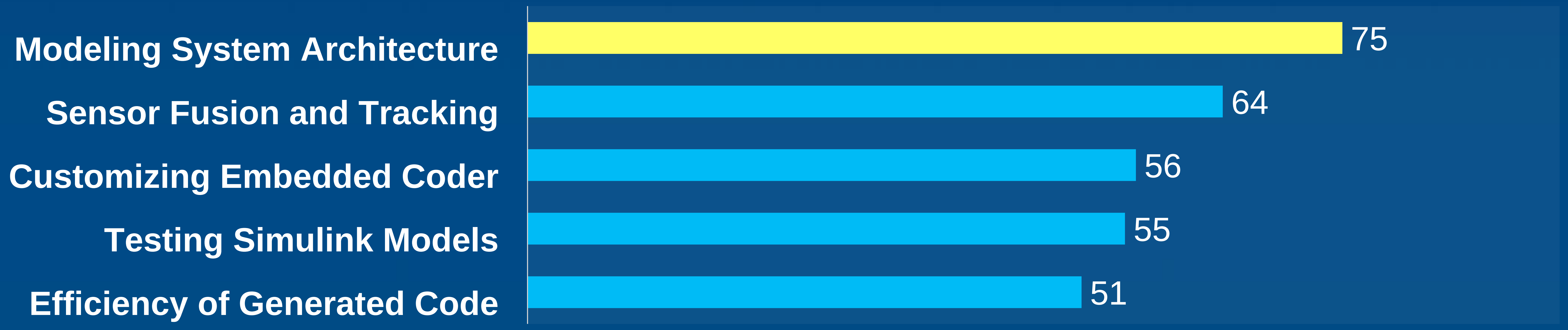


```
loop_ub = bw_a_filled->size[0] - 2;
b_loop_ub = bw_a_filled->size[1] - 2;
i0 = bw_filled->size[0] * bw_filled->size[1];
bw_filled->size[0] = loop_ub + 1;
bw_filled->size[1] = b_loop_ub + 1;
emxEnsureCapacity((emxArray__common *)bw_filled, i0, (i
emxFree_boolean_T(&b_bw_b);
for (i0 = 0; i0 <= b_loop_ub; i0++) {
  for (i1 = 0; i1 <= loop_ub; i1++) {
    bw_filled->data[i1 + bw_filled->size[0] * i0] = (bw_
      bw_a_filled->size[0] * (1 + i0)) + 1] || bw_b_fil
      bw_b_filled->size[0] * i0) + 1] || bw_c_filled->da
      bw_c_filled->size[0] * i0] || bw_b->data[i1 + bw_
  }
}
```

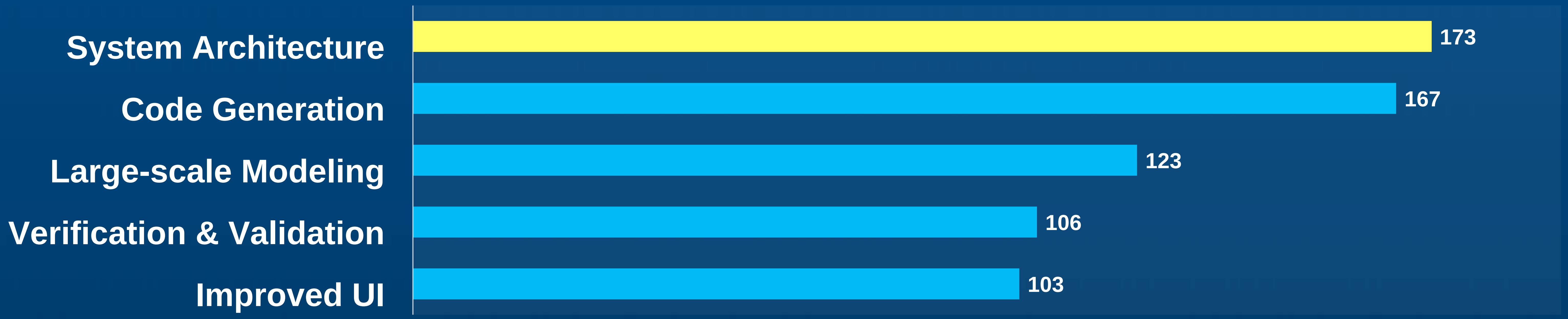
Components

System architecture is the #1 topic

Breakout Topic Requests (2018)

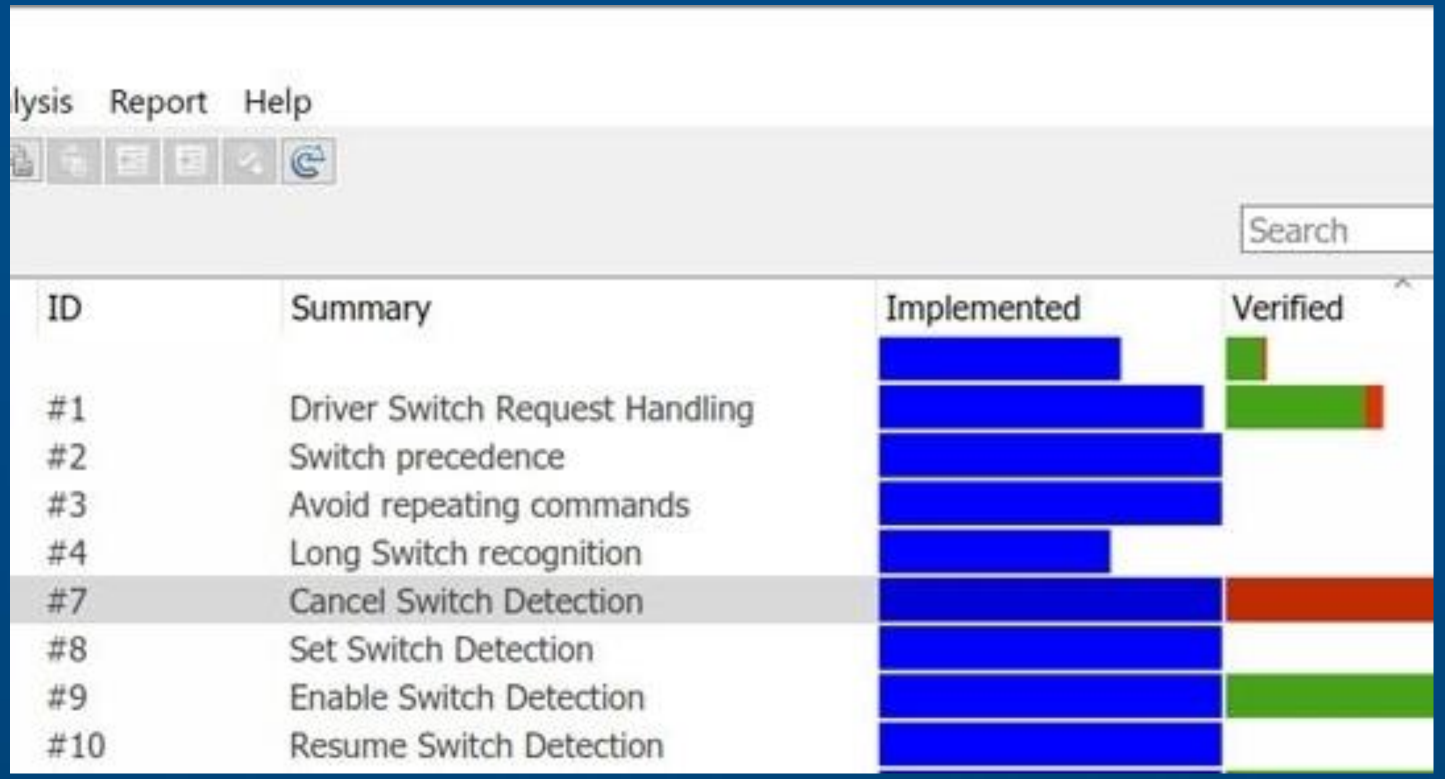


Feature Prioritization (2017)



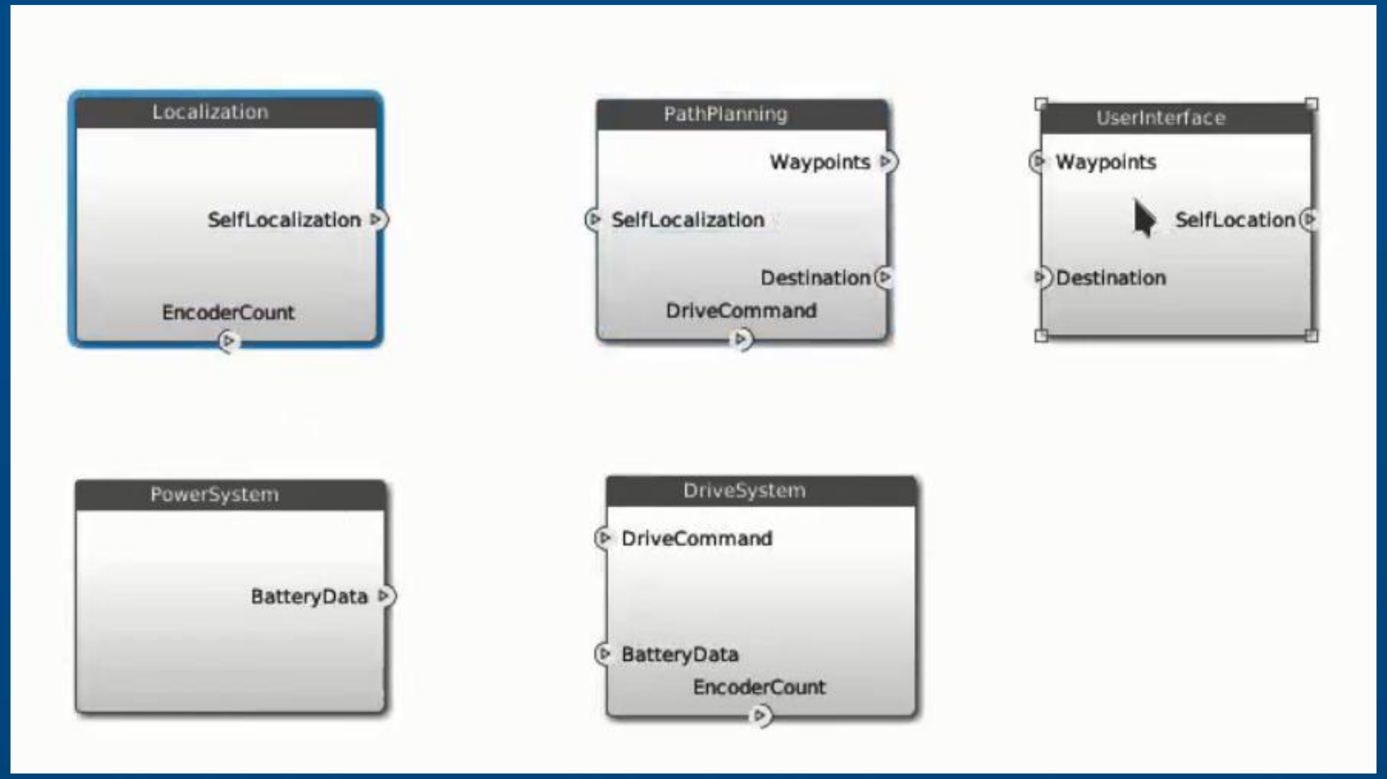
Systems engineering

Requirements

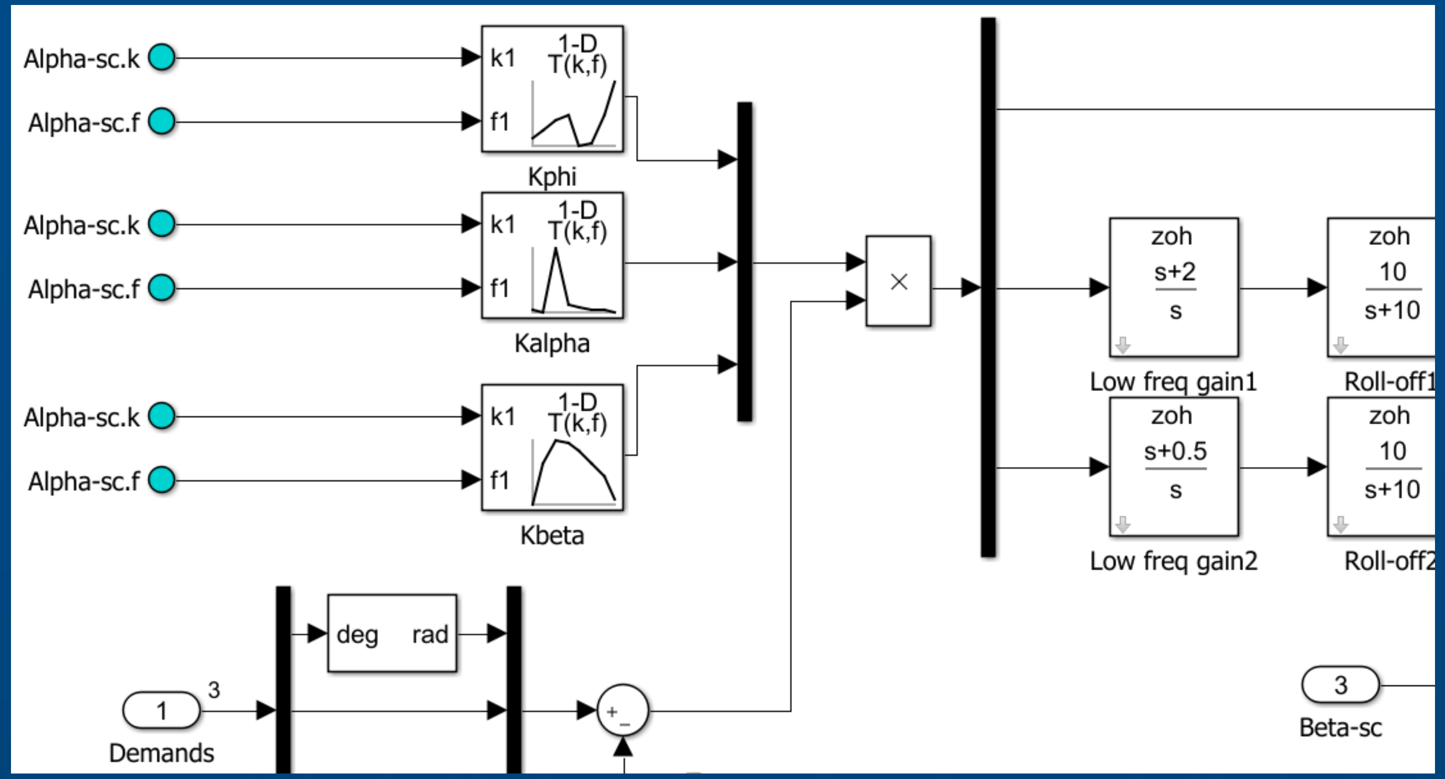


ID	Summary	Implemented	Verified
		🟦	🟩
#1	Driver Switch Request Handling	🟦	🟩🟥
#2	Switch precedence	🟦	
#3	Avoid repeating commands	🟦	
#4	Long Switch recognition	🟦	
#7	Cancel Switch Detection	🟦	🟥
#8	Set Switch Detection	🟦	
#9	Enable Switch Detection	🟦	🟩
#10	Resume Switch Detection	🟦	

Systems



Components

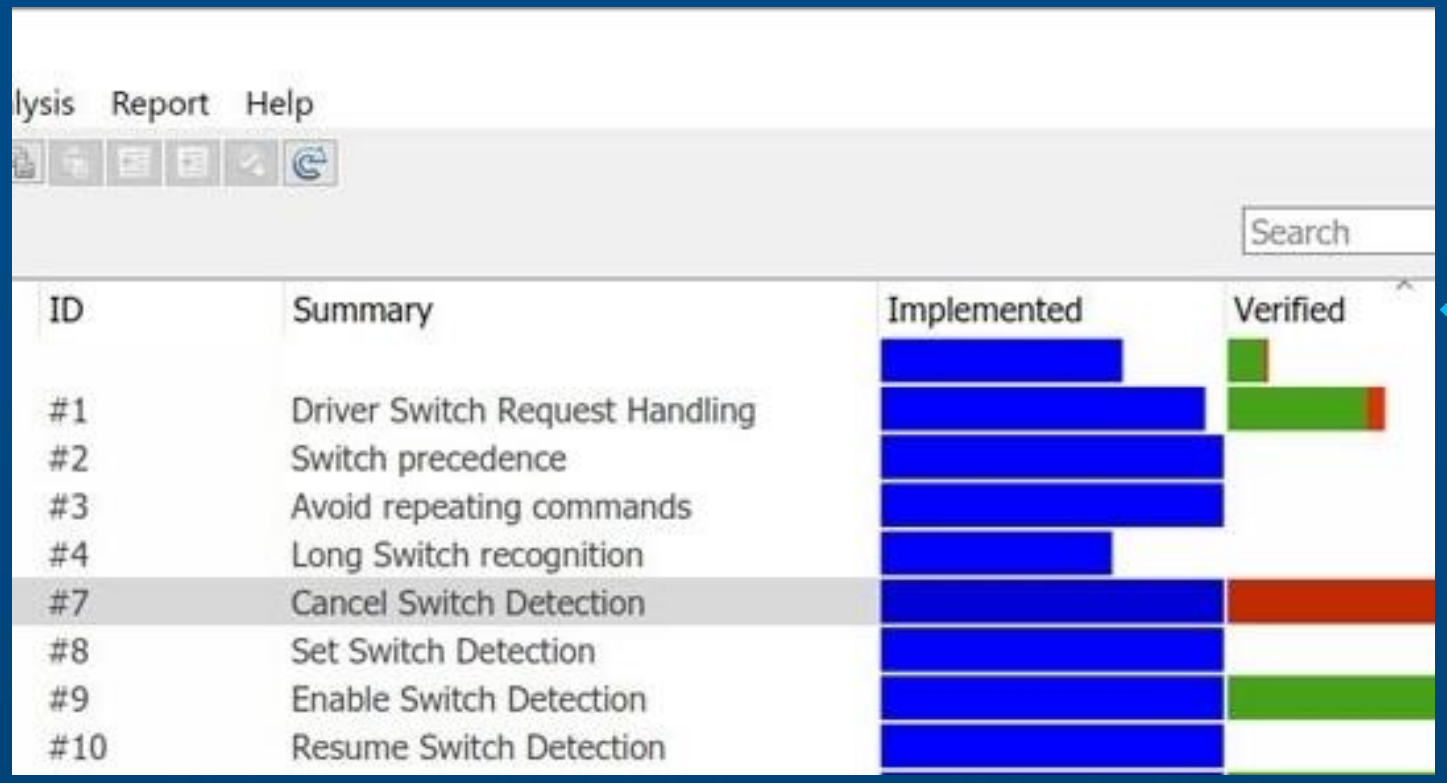


Systems engineering

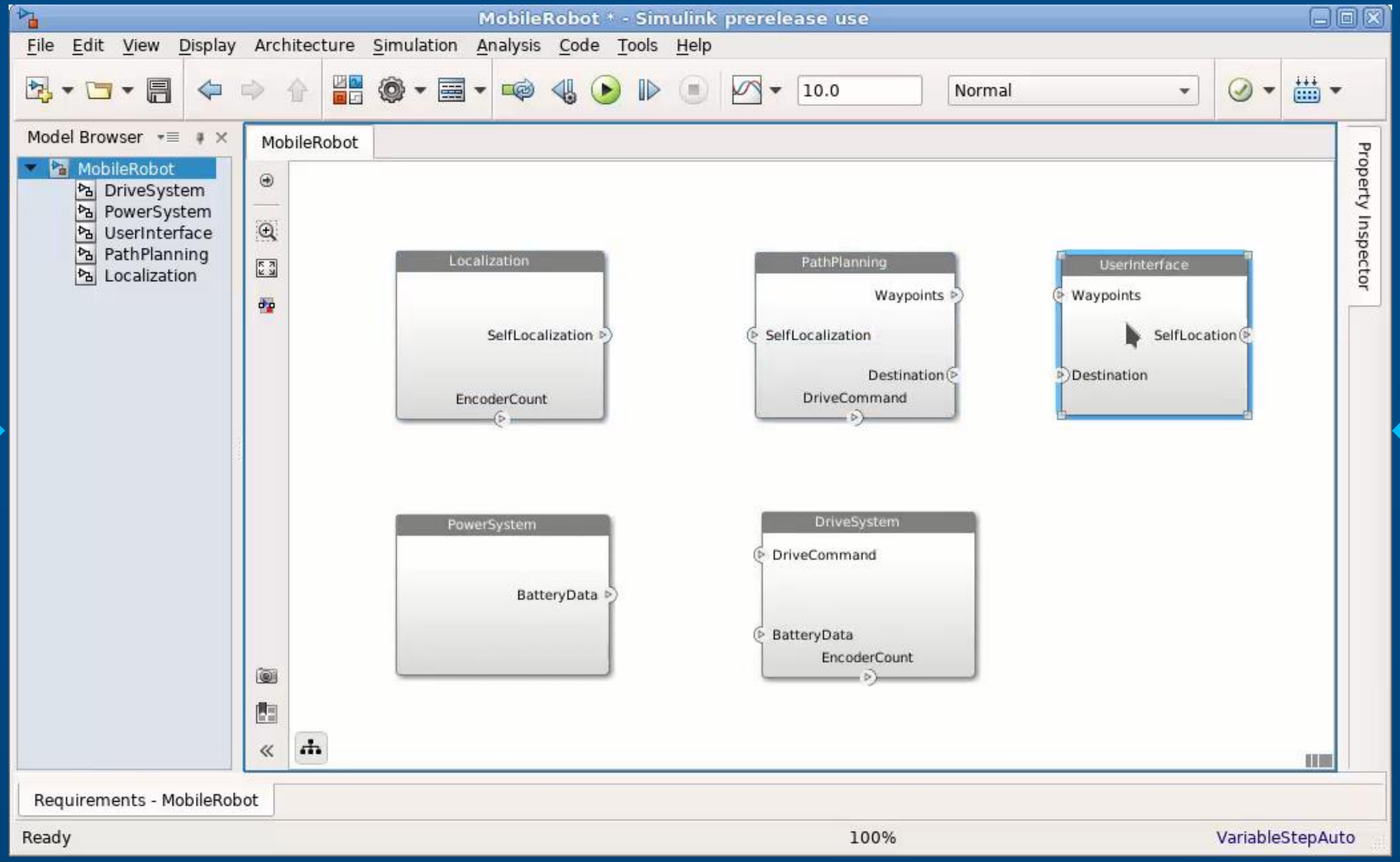
R2019a

System Composer

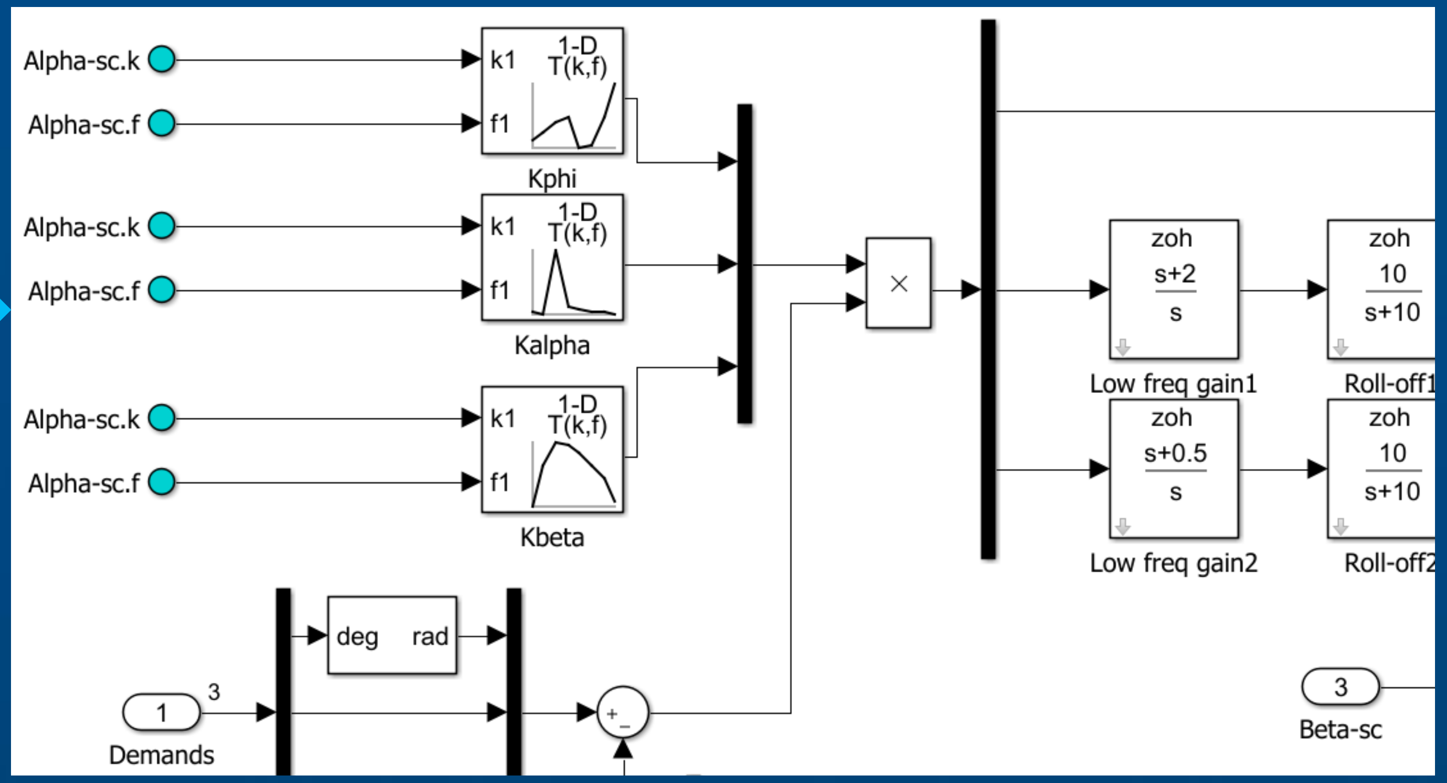
Requirements



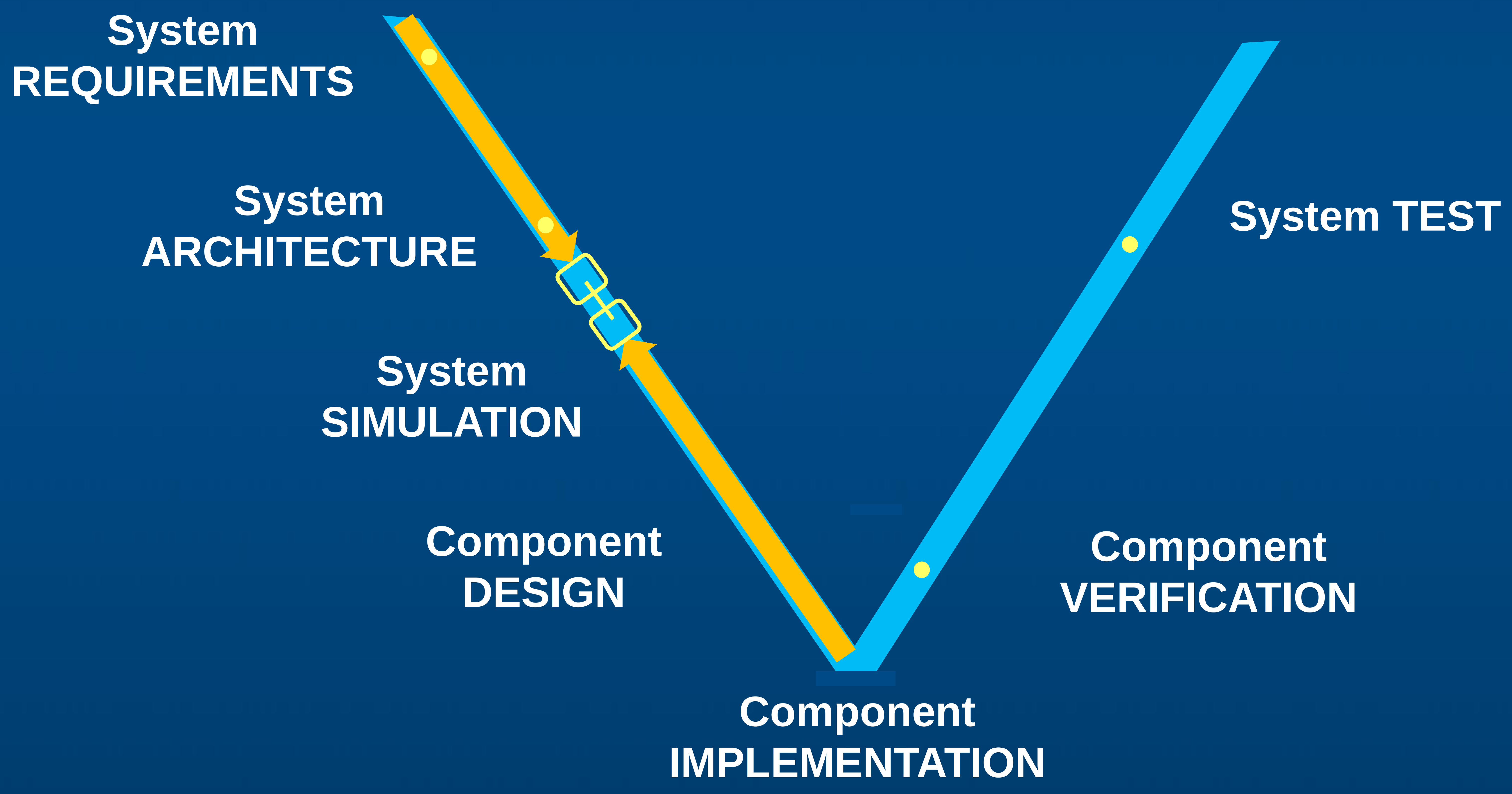
ID	Summary	Implemented	Verified
		🟦	🟩
#1	Driver Switch Request Handling	🟦	🟩🟥
#2	Switch precedence	🟦	
#3	Avoid repeating commands	🟦	
#4	Long Switch recognition	🟦	
#7	Cancel Switch Detection	🟦	🟥
#8	Set Switch Detection	🟦	
#9	Enable Switch Detection	🟦	🟩
#10	Resume Switch Detection	🟦	



Components

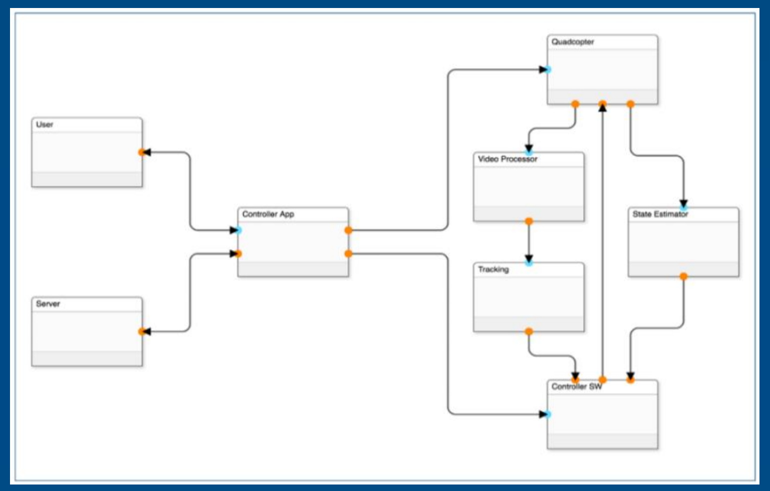


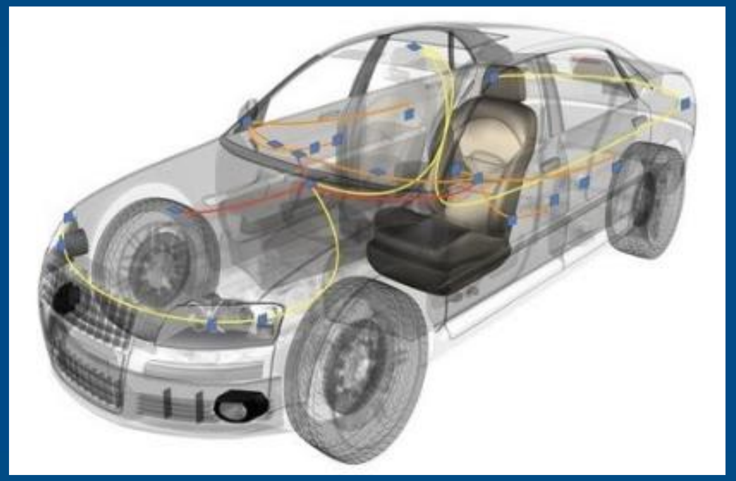
Linking top-down and bottom-up workflows



Types of models

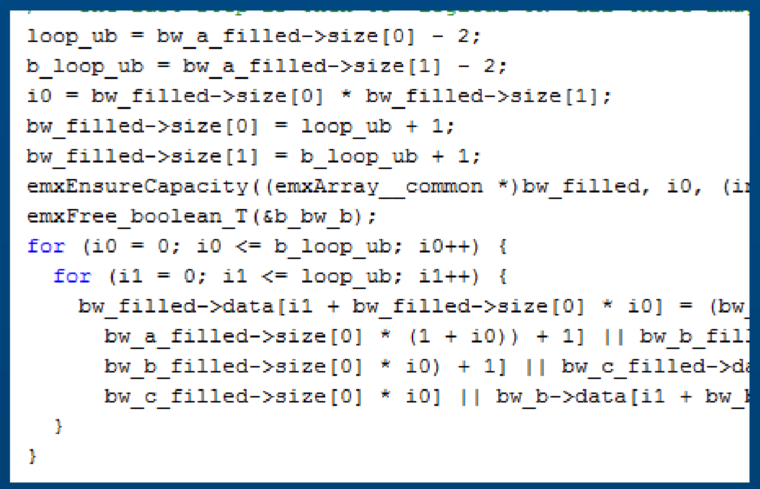
Systems





Software

Physics



```
loop_ub = bw_a_filled->size[0] - 2;
b_loop_ub = bw_a_filled->size[1] - 2;
i0 = bw_filled->size[0] * bw_filled->size[1];
bw_filled->size[0] = loop_ub + 1;
bw_filled->size[1] = b_loop_ub + 1;
emxEnsureCapacity((emxArray__common *)bw_filled, i0, (i
emxFree_boolean_T(&b_bw_b);
for (i0 = 0; i0 <= b_loop_ub; i0++) {
  for (i1 = 0; i1 <= loop_ub; i1++) {
    bw_filled->data[i1 + bw_filled->size[0] * i0] = (bw_
      bw_a_filled->size[0] * (1 + i0)) + 1] || bw_b_fil
      bw_b_filled->size[0] * i0) + 1] || bw_c_filled->da
      bw_c_filled->size[0] * i0] || bw_b->data[i1 + bw_b
  }
}
```

Components

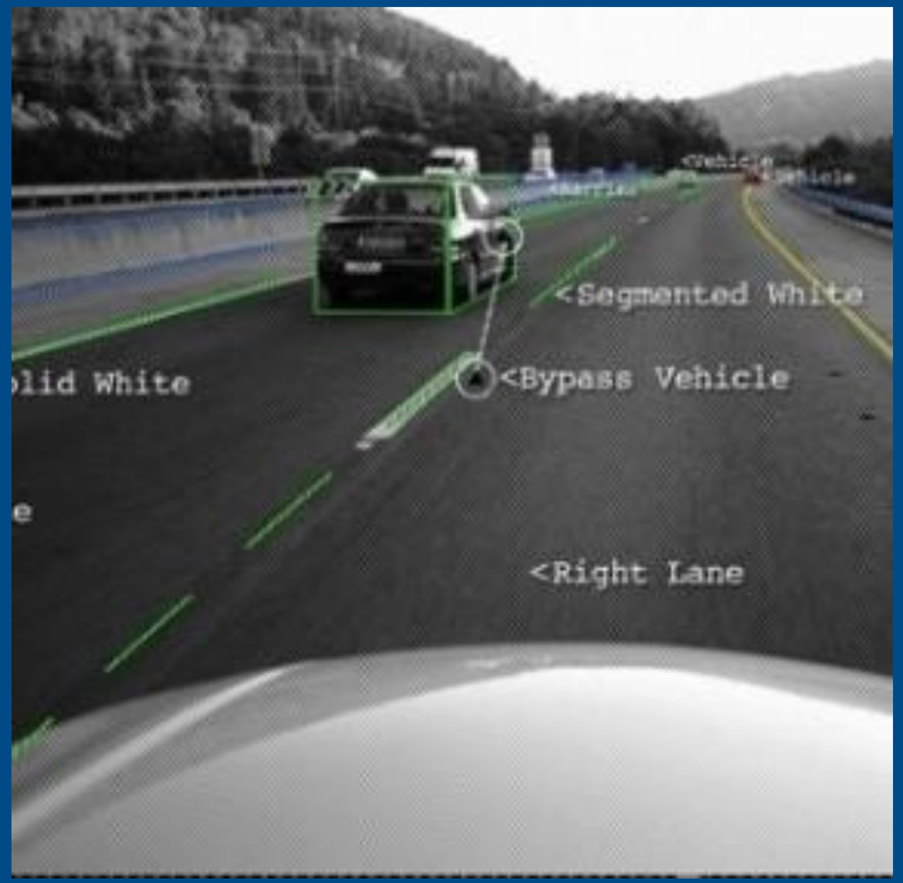
Deep solutions

Controls

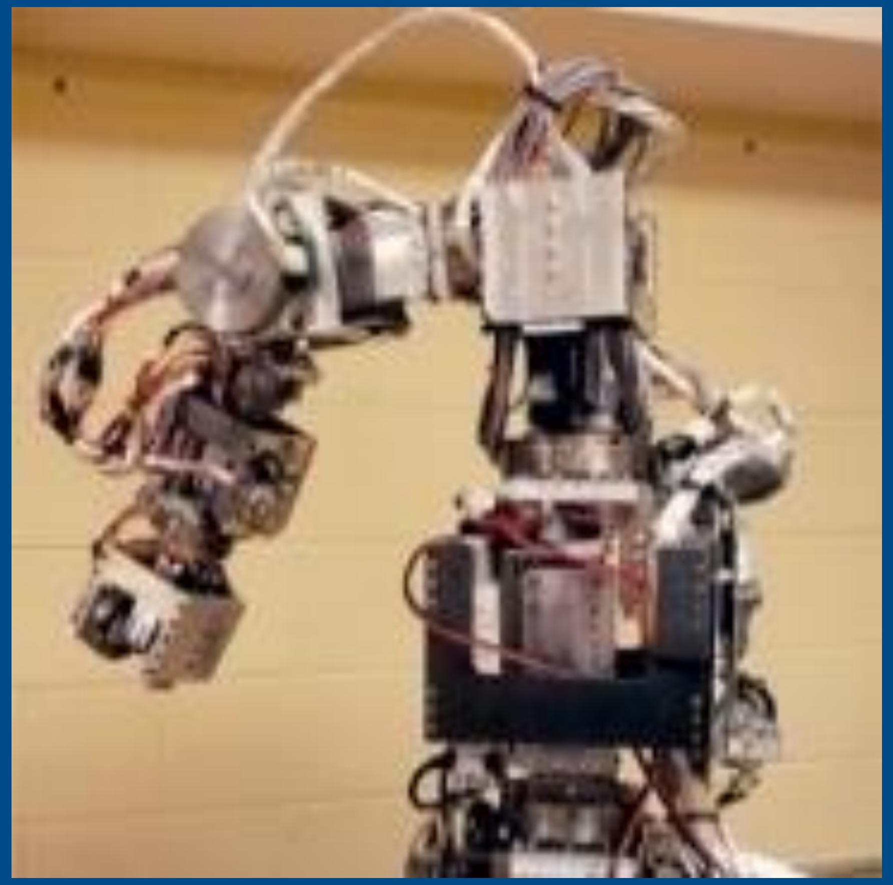
Signal Processing

Wireless

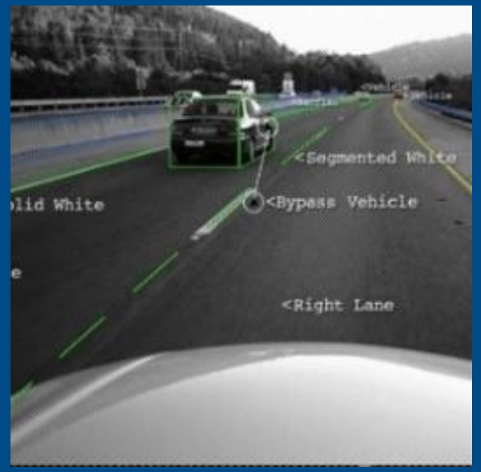
Vision

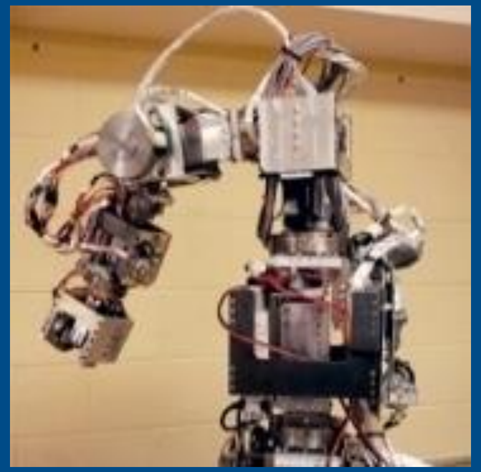


Robotics

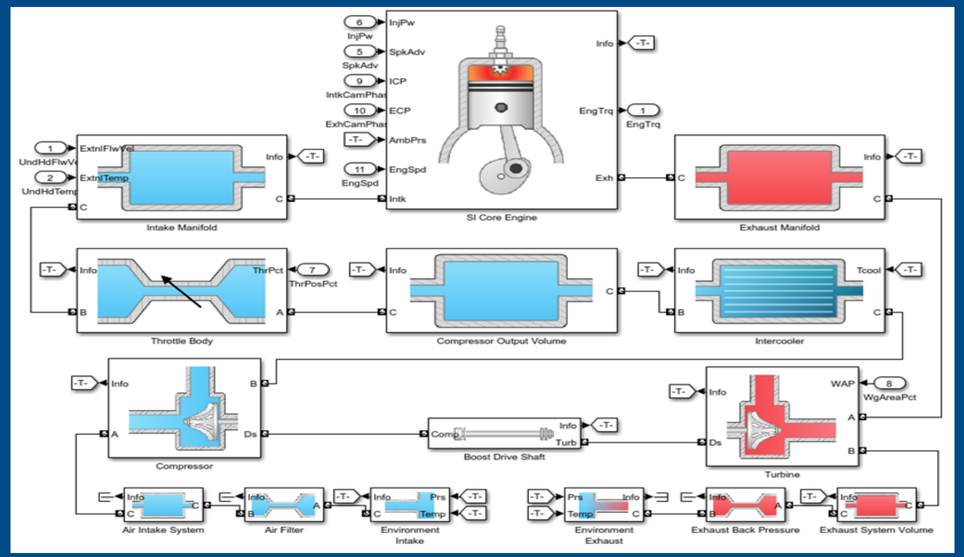


Deep solutions



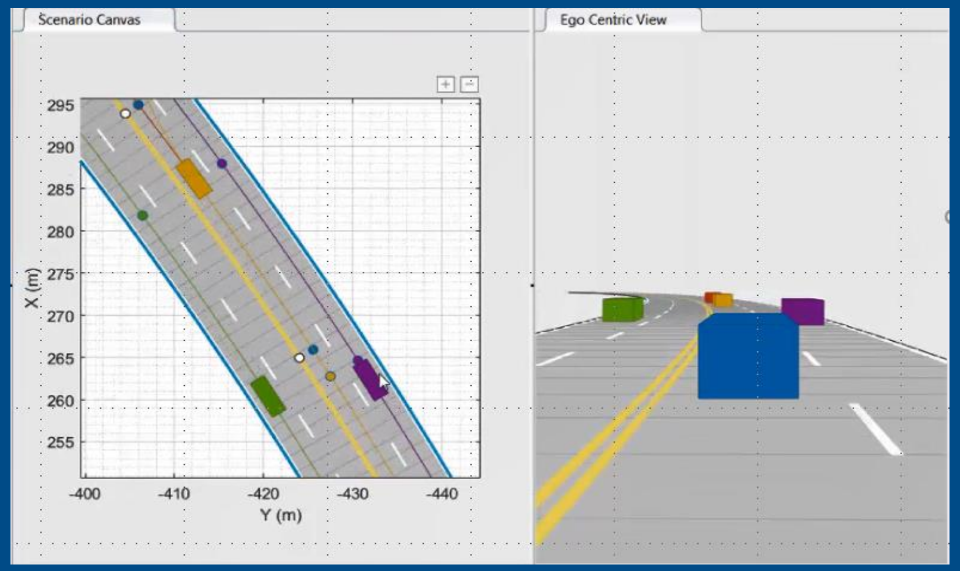


Automotive Products

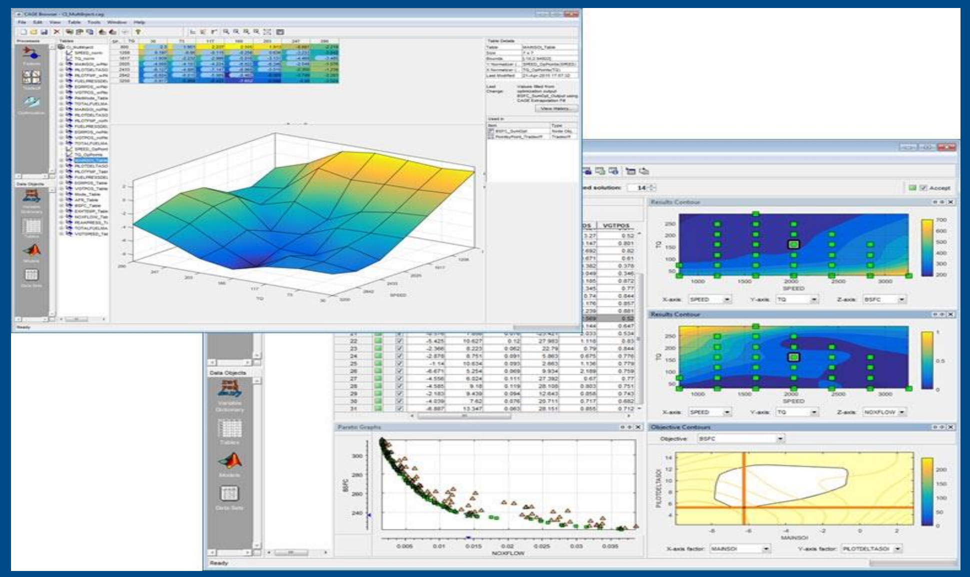


Powertrain

Vehicle

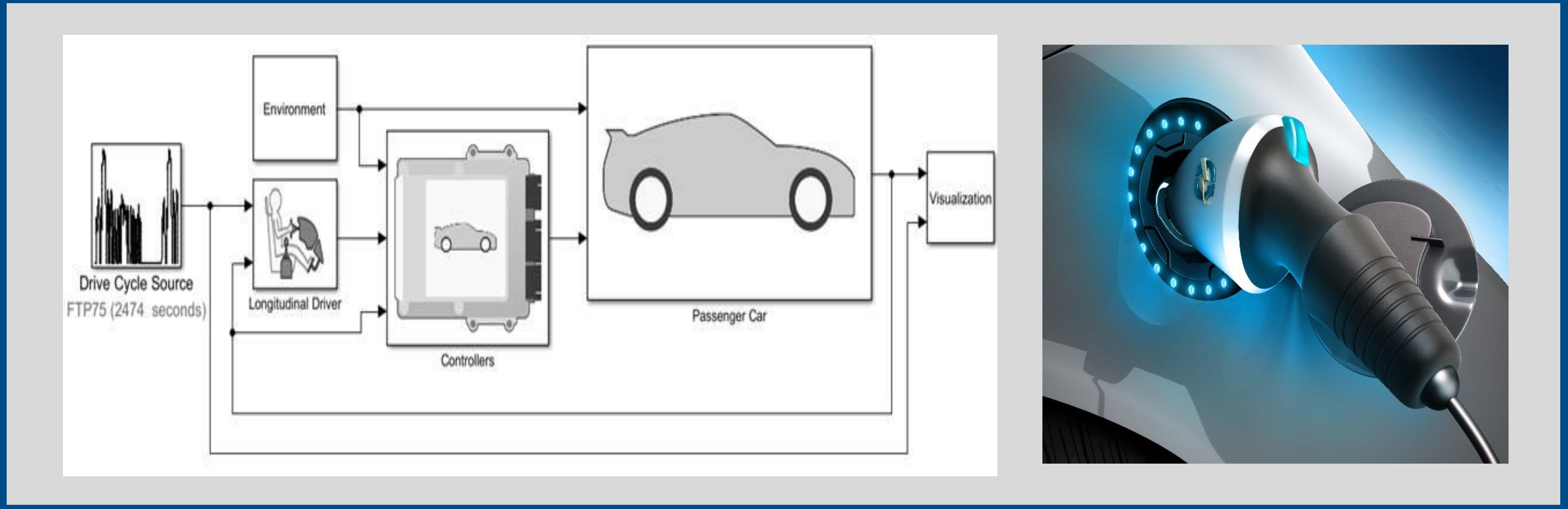


Automated Driving

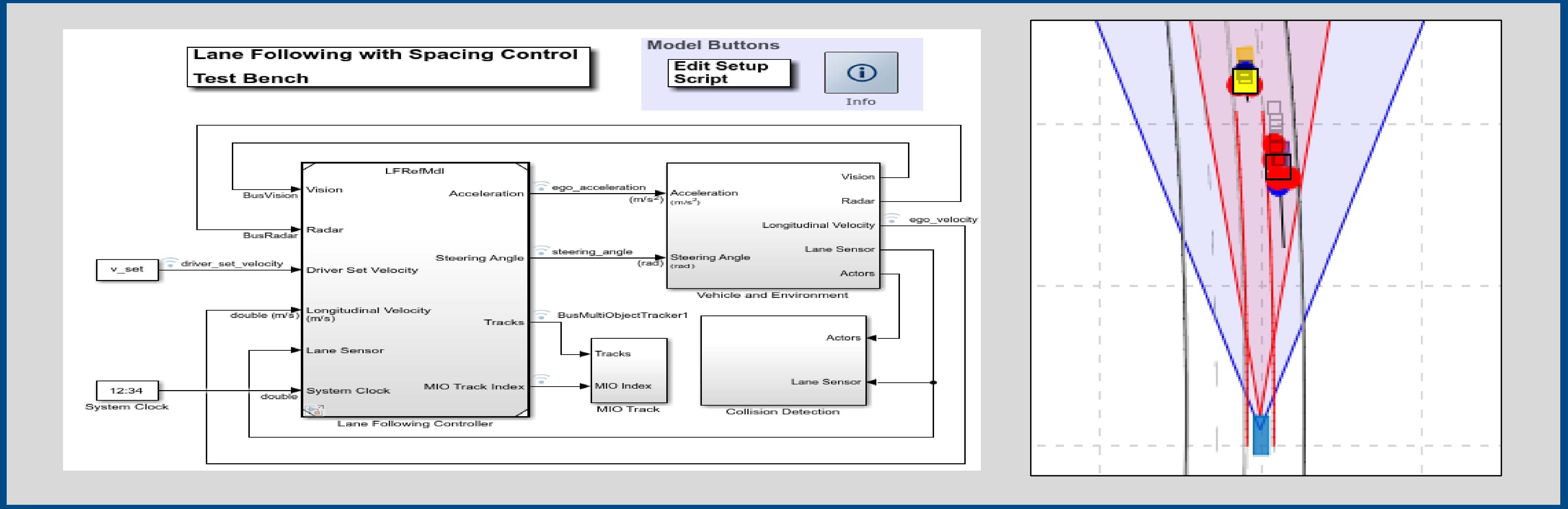


Calibration

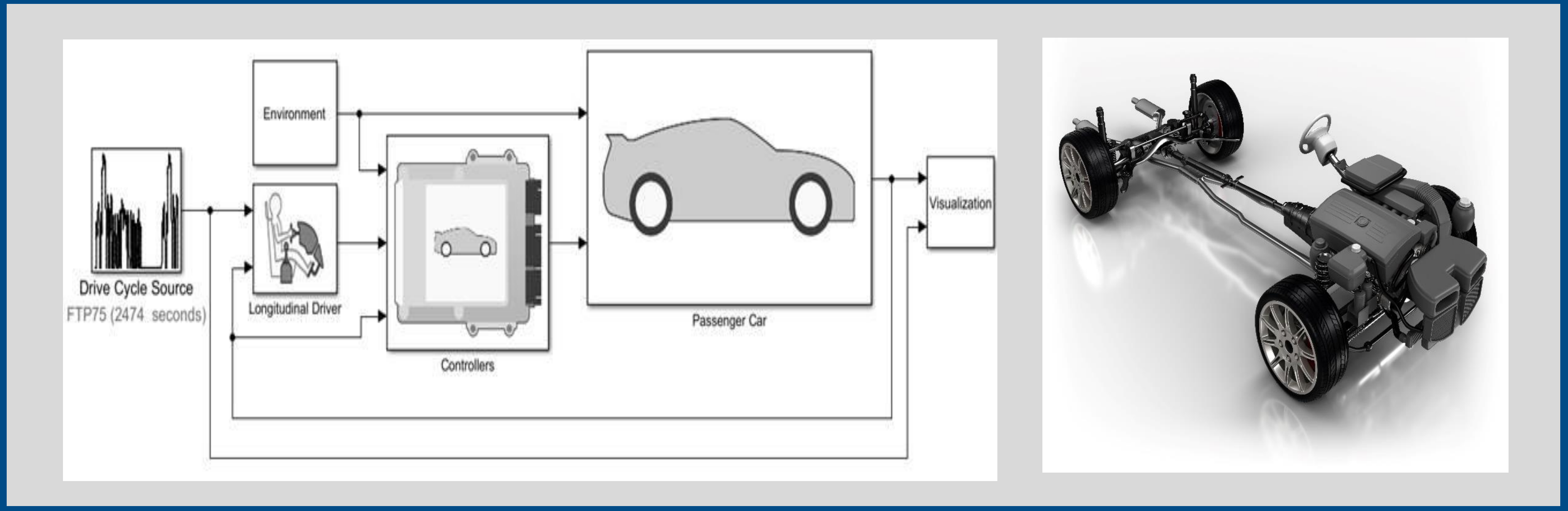
Automotive Reference Applications



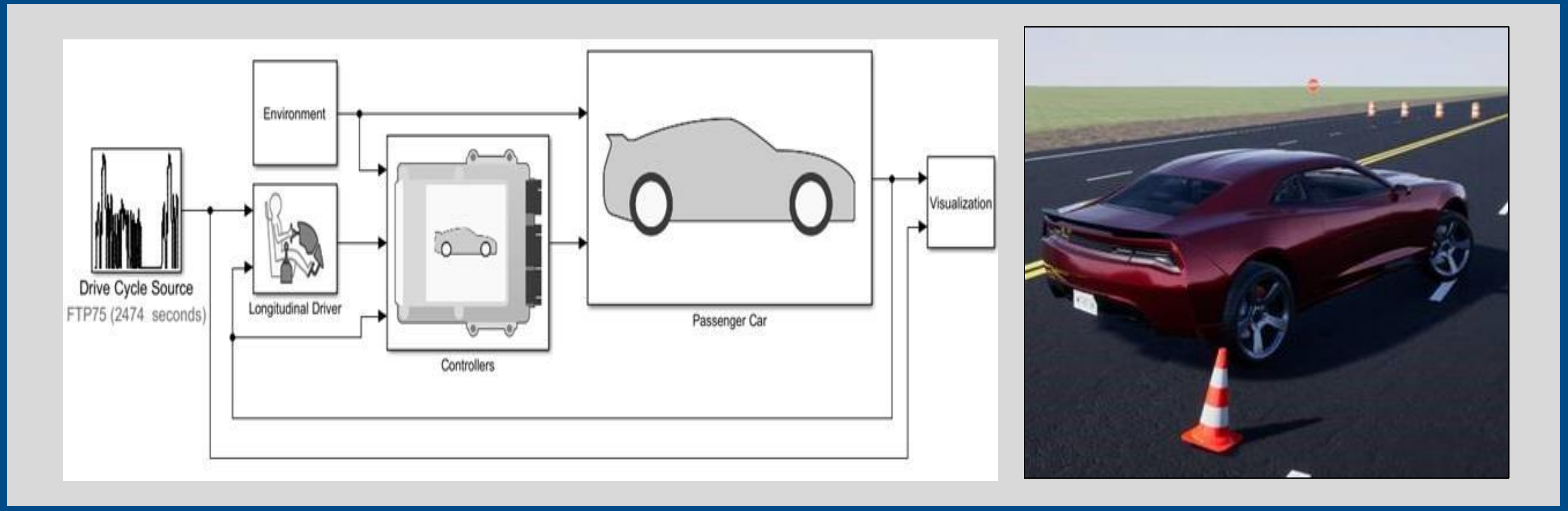
Pure EV



Lane Keeping Assist



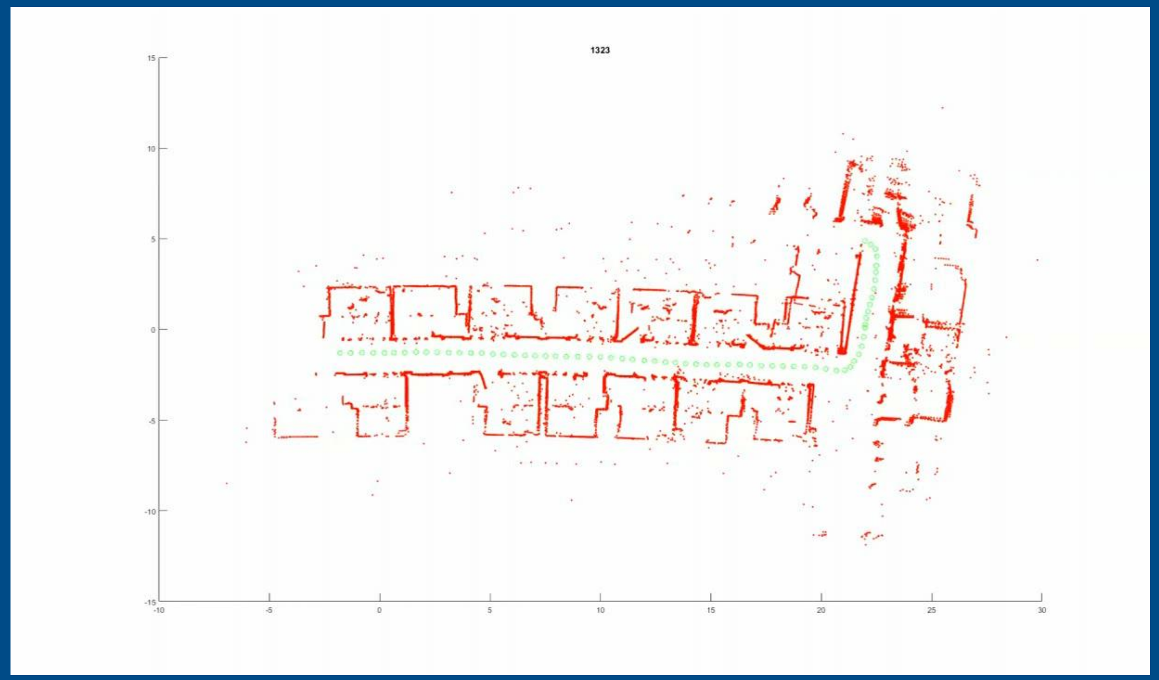
Hybrid Powertrain



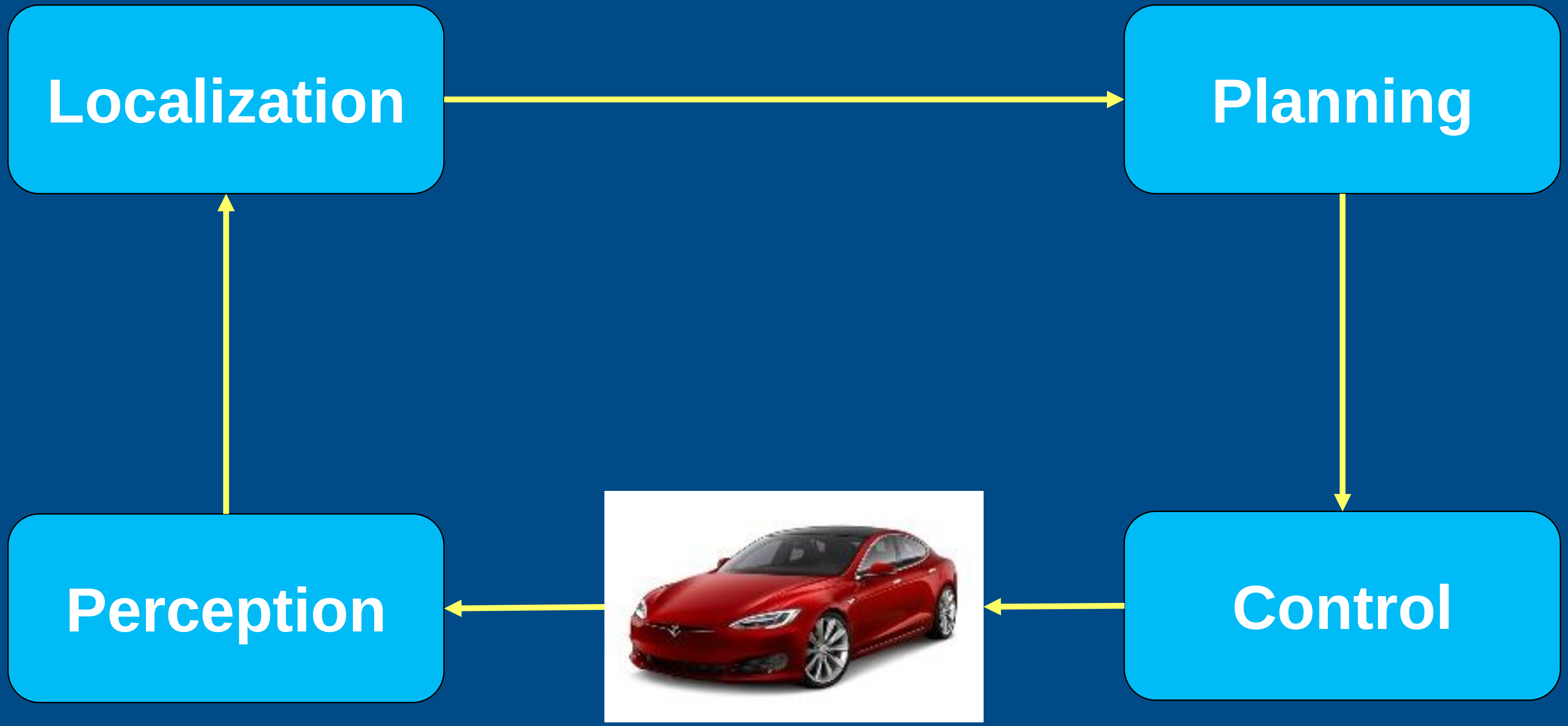
Car Vehicle Dynamics

Deep solutions for autonomous systems

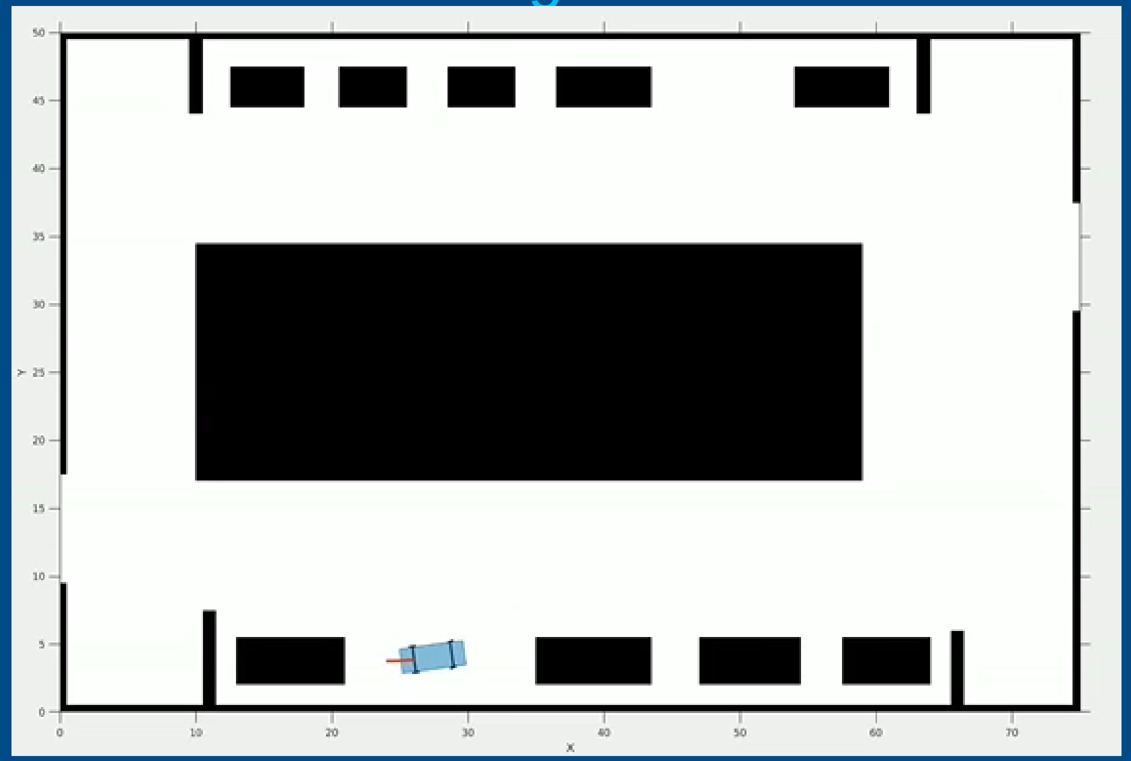
SLAM (18a)
Robotics System Toolbox



Semantic Segmentation (17b)
Automated Driving
System Toolbox

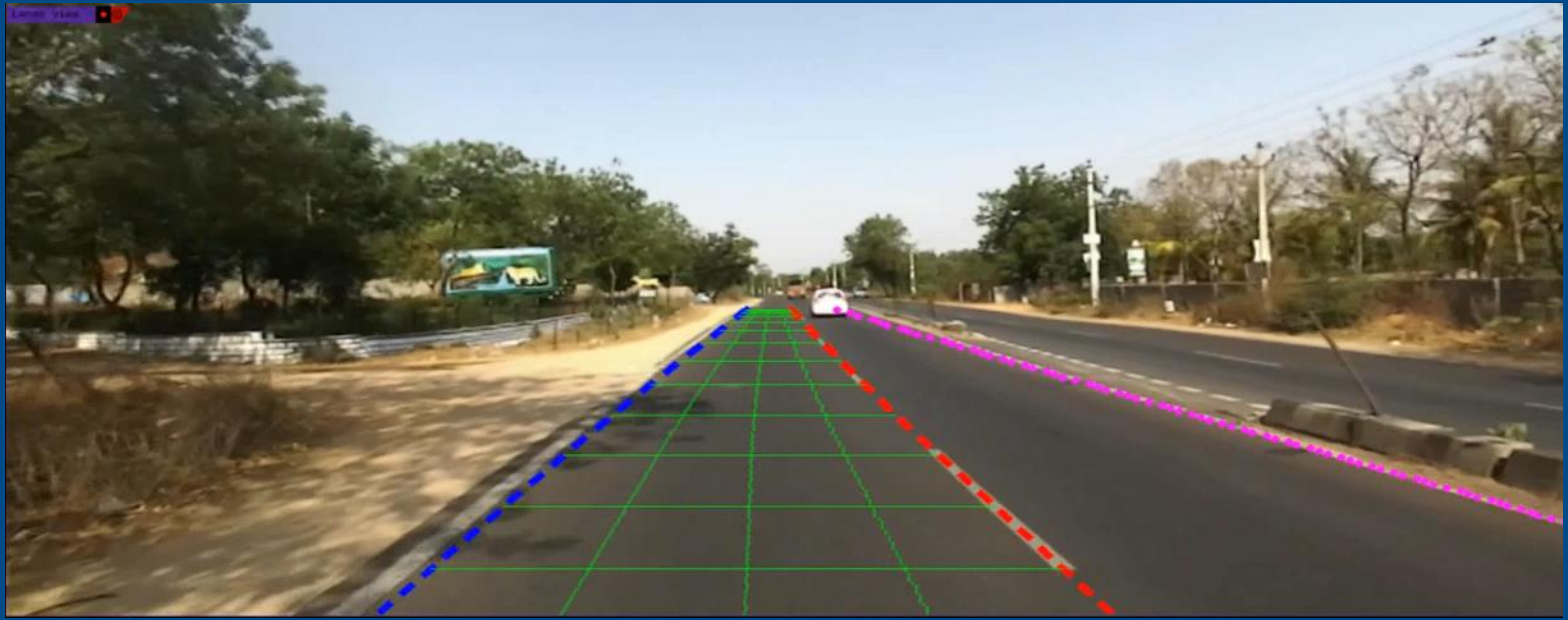


Path Planning (19a)
Automated Driving Toolbox



Adaptive Cruise Control (17a)
Automated Driving
System Toolbox

Deep solutions for autonomous systems

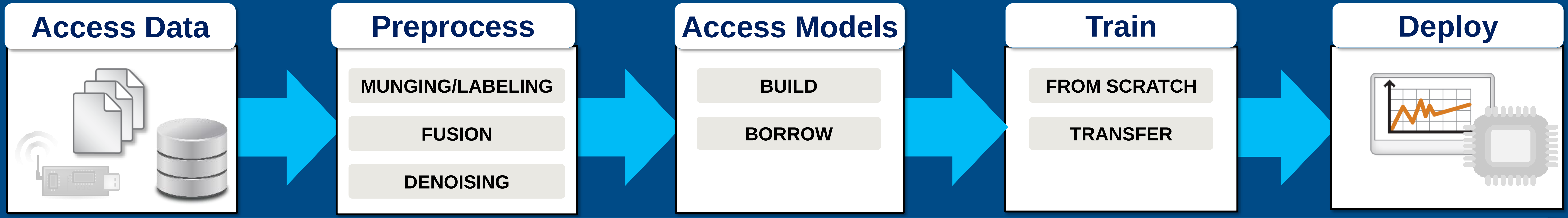


Lane Keep Assist
Model Predictive Control

Automatic Emergency Braking
Automated Driving Toolbox

MODELING & SIMULATION

MATLAB Workflow for Deep Learning:



Access Data

Preprocess

- MUNGING/LABELING
- FUSION
- DENOISING

Access Models

- BUILD
- BORROW

Train

- FROM SCRATCH
- TRANSFER

Deploy

Deep Learning Toolbox

Create, analyze, and train deep learning networks

Interoperability with open source networks

ONNX

PyTorch mxnet TensorFlow

Deep Network Designer App

Inference performance

NVIDIA

Domain-specific workflow support

Ground truth labeling apps for:

- Video
- Audio
- application-specific datastores

Network training performance

NVIDIA GPU CLOUD

Azure

amazon web services™

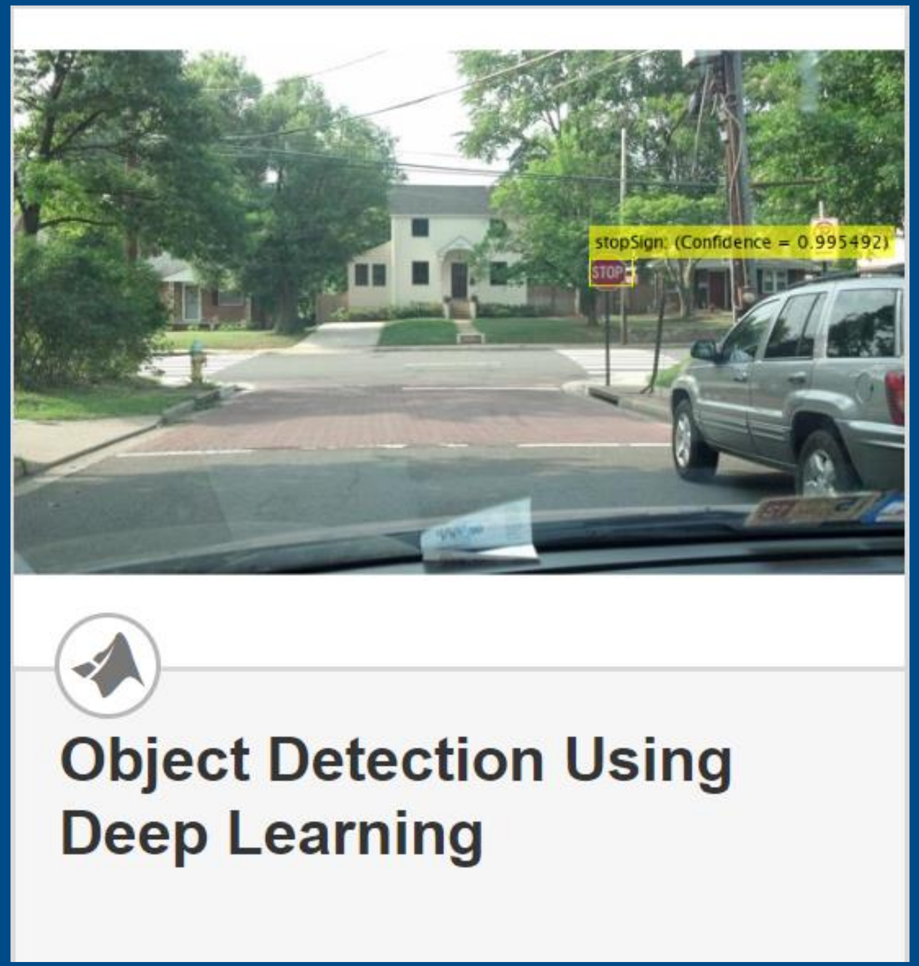
Deployment support

intel

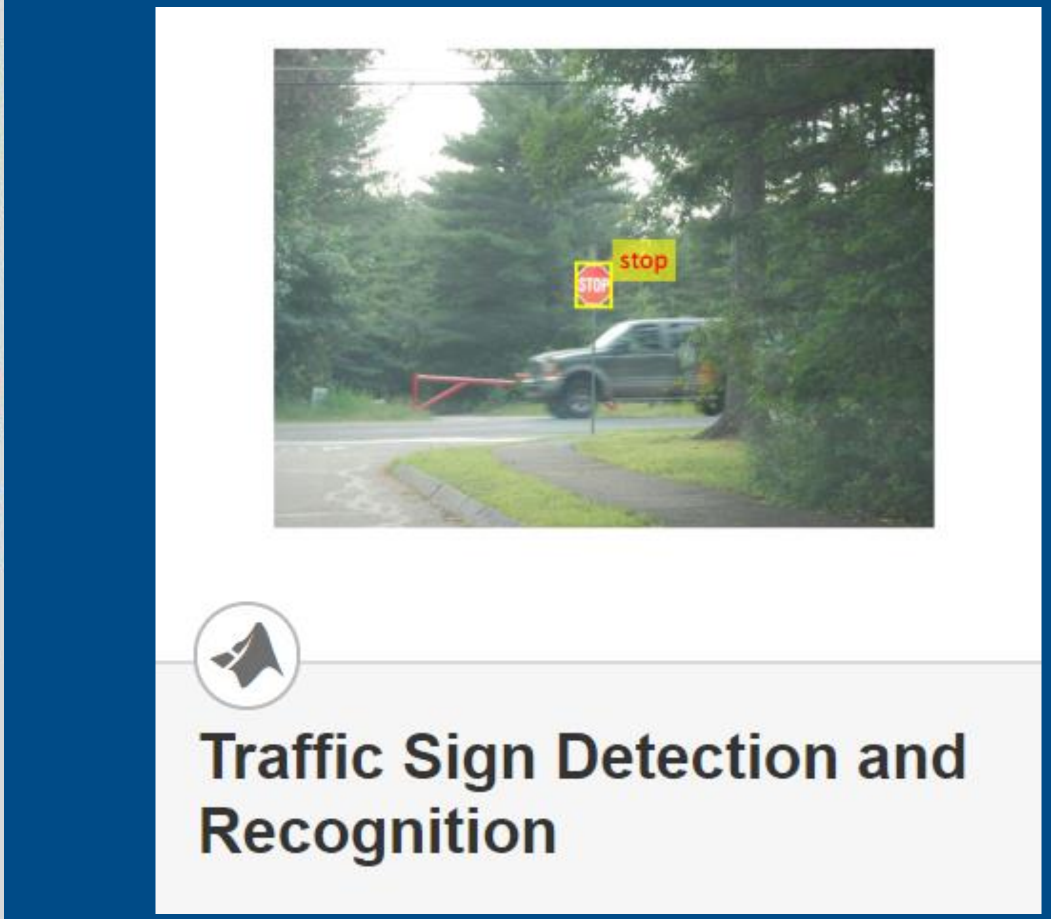
ARM

Artificial Intelligence for your applications

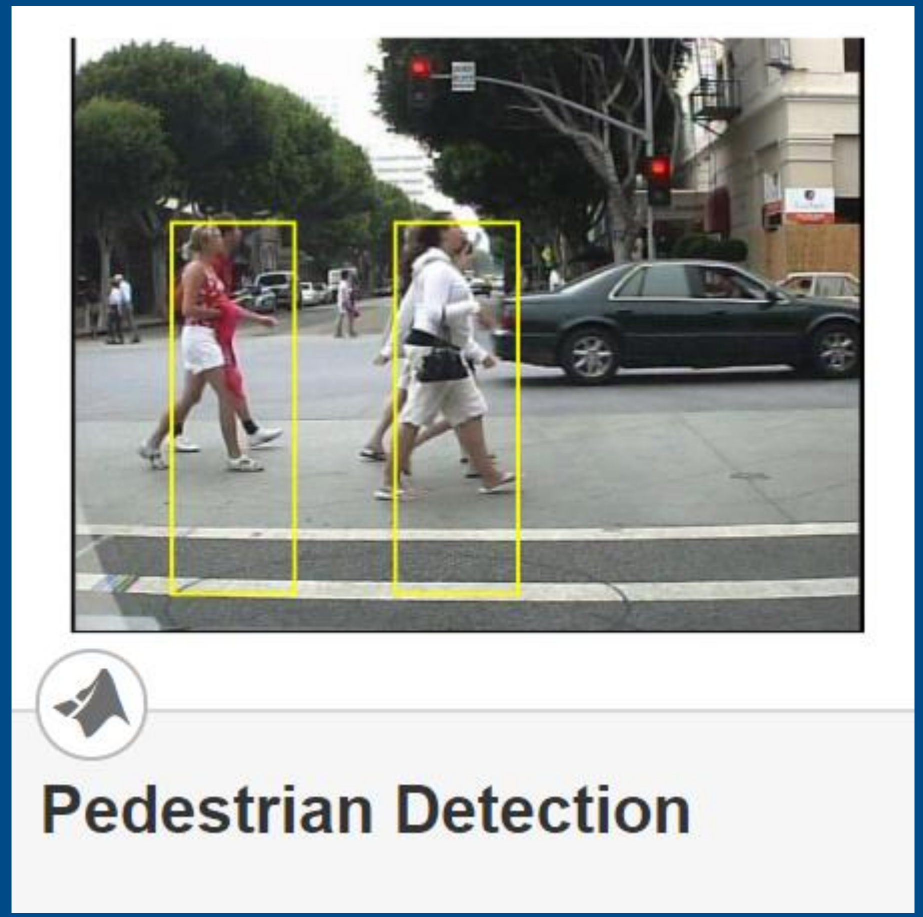
- Application examples



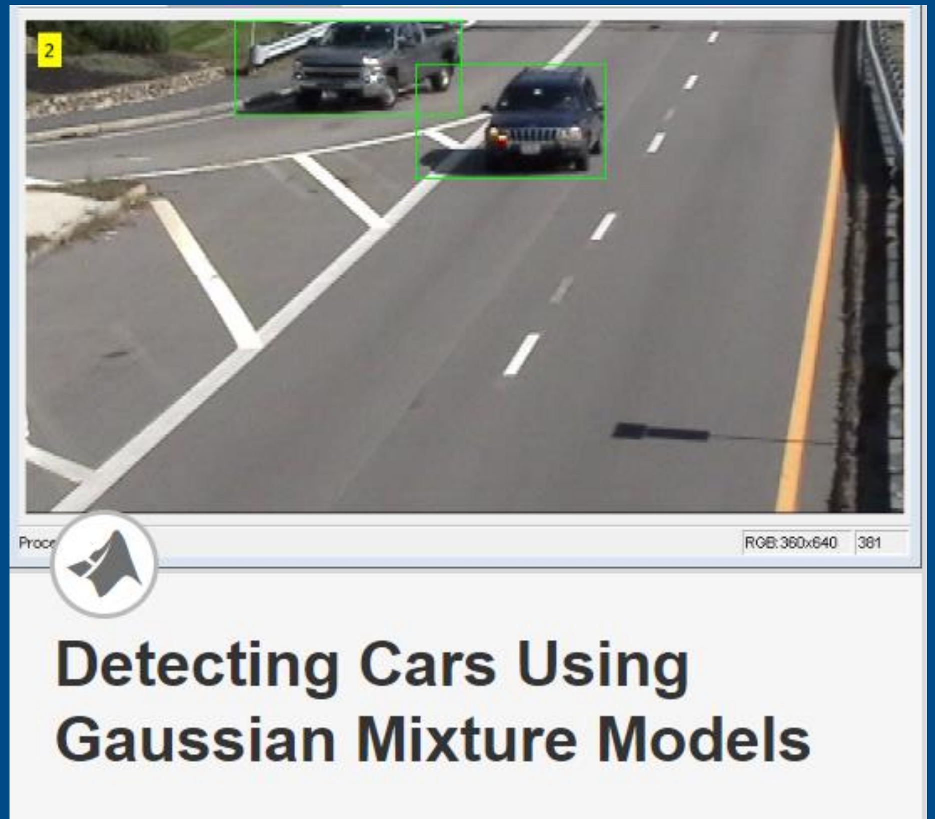
Object Detection Using Deep Learning



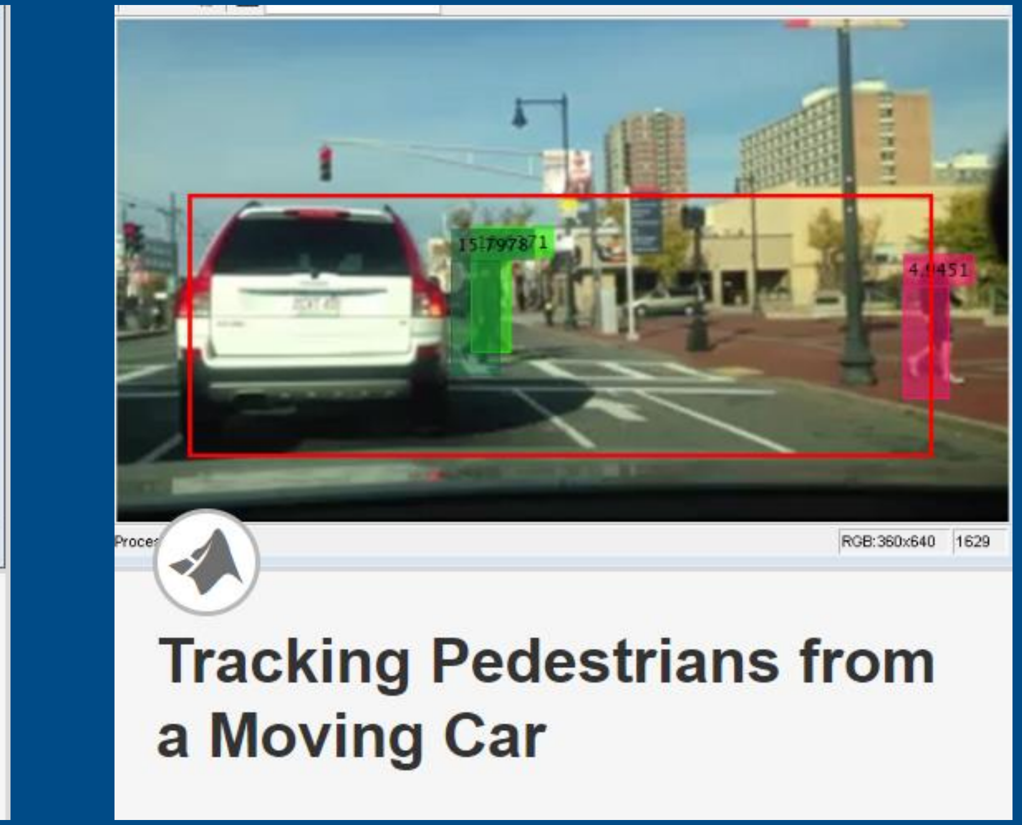
Traffic Sign Detection and Recognition



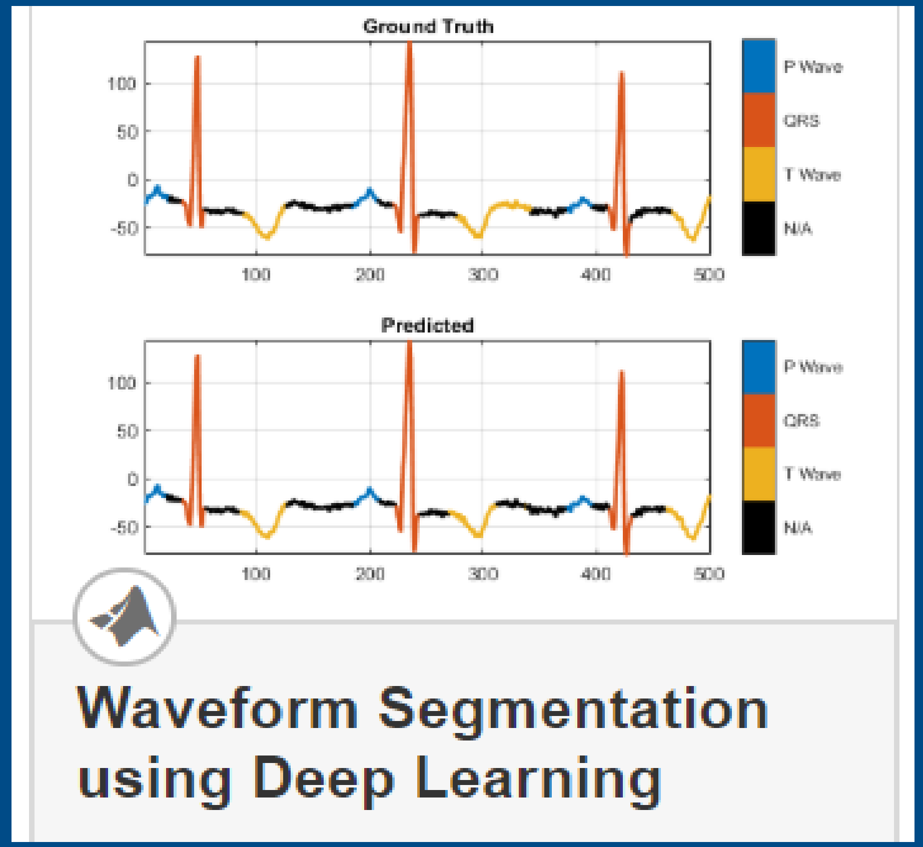
Pedestrian Detection



Detecting Cars Using Gaussian Mixture Models



Tracking Pedestrians from a Moving Car

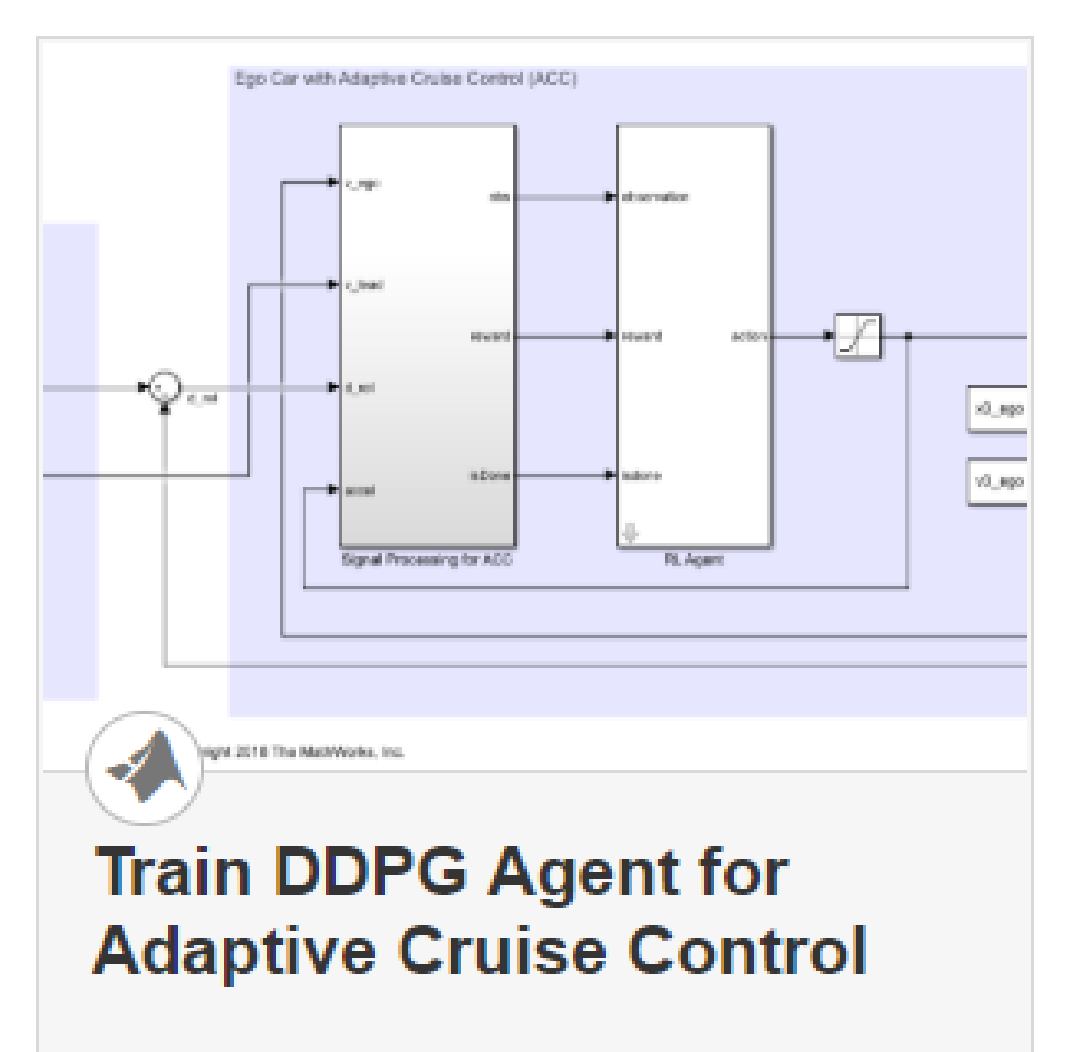


Waveform Segmentation using Deep Learning

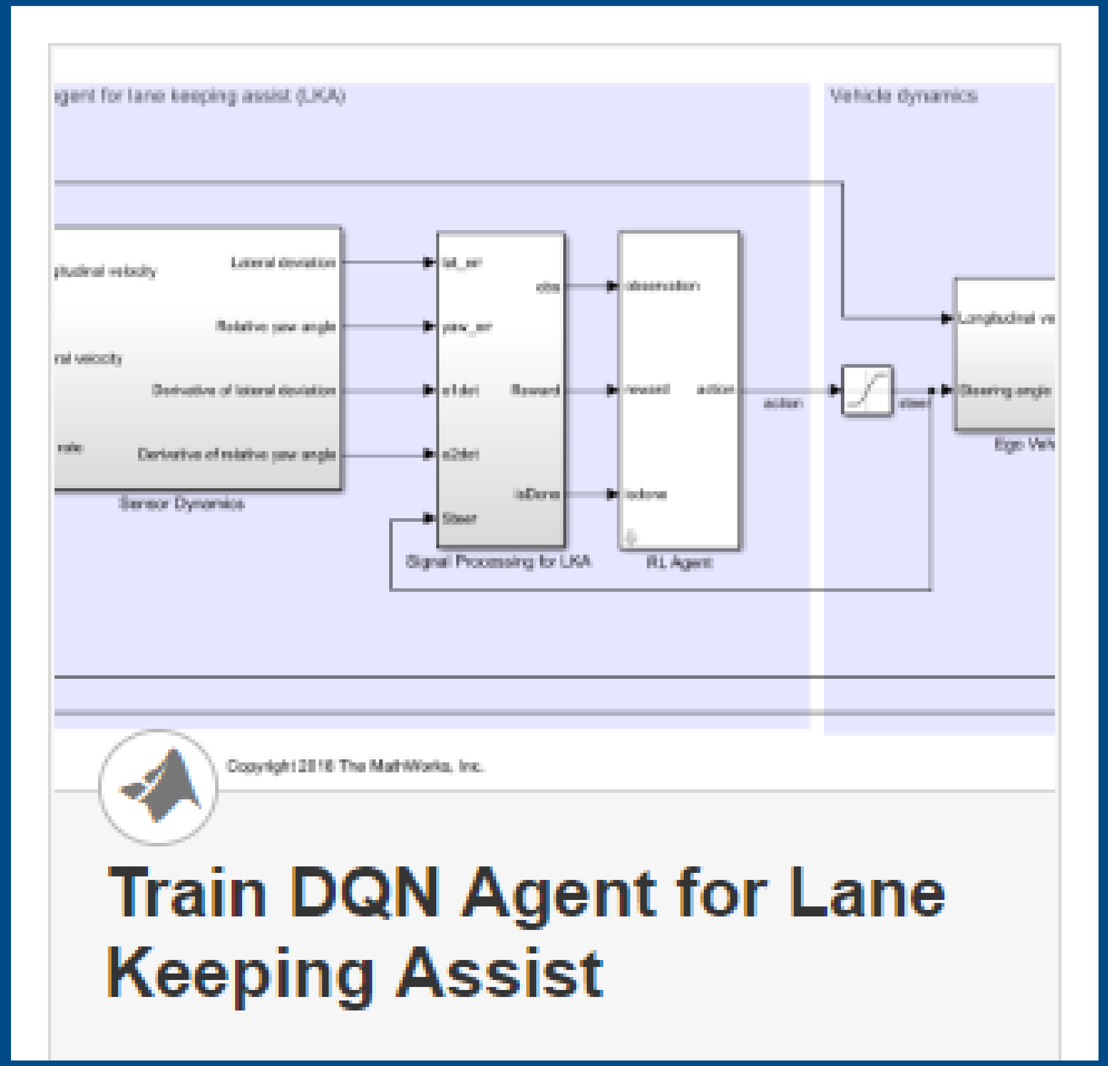
Artificial Intelligence for your applications

R2019a

- Application examples
- Control design


Ego Car with Adaptive Cruise Control (ACC)
v_ego
v_lead
d_rel
accel
obs
reward
isDone
Signal Processing for ACC
observation
reward
isdone
action
RL Agent
x0_ego
v0_ego
ight 2018 The MathWorks, Inc.

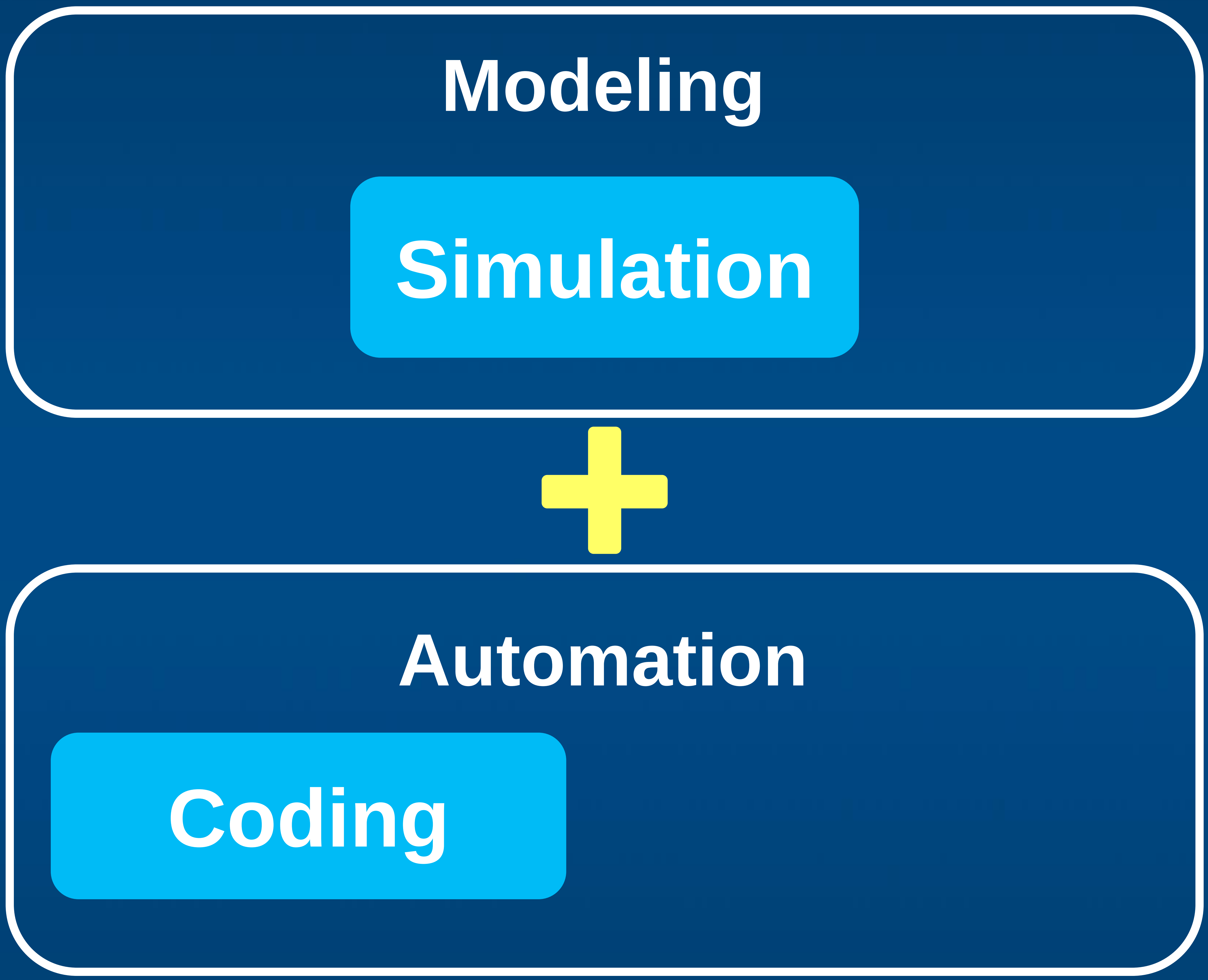
Train DDPG Agent for Adaptive Cruise Control



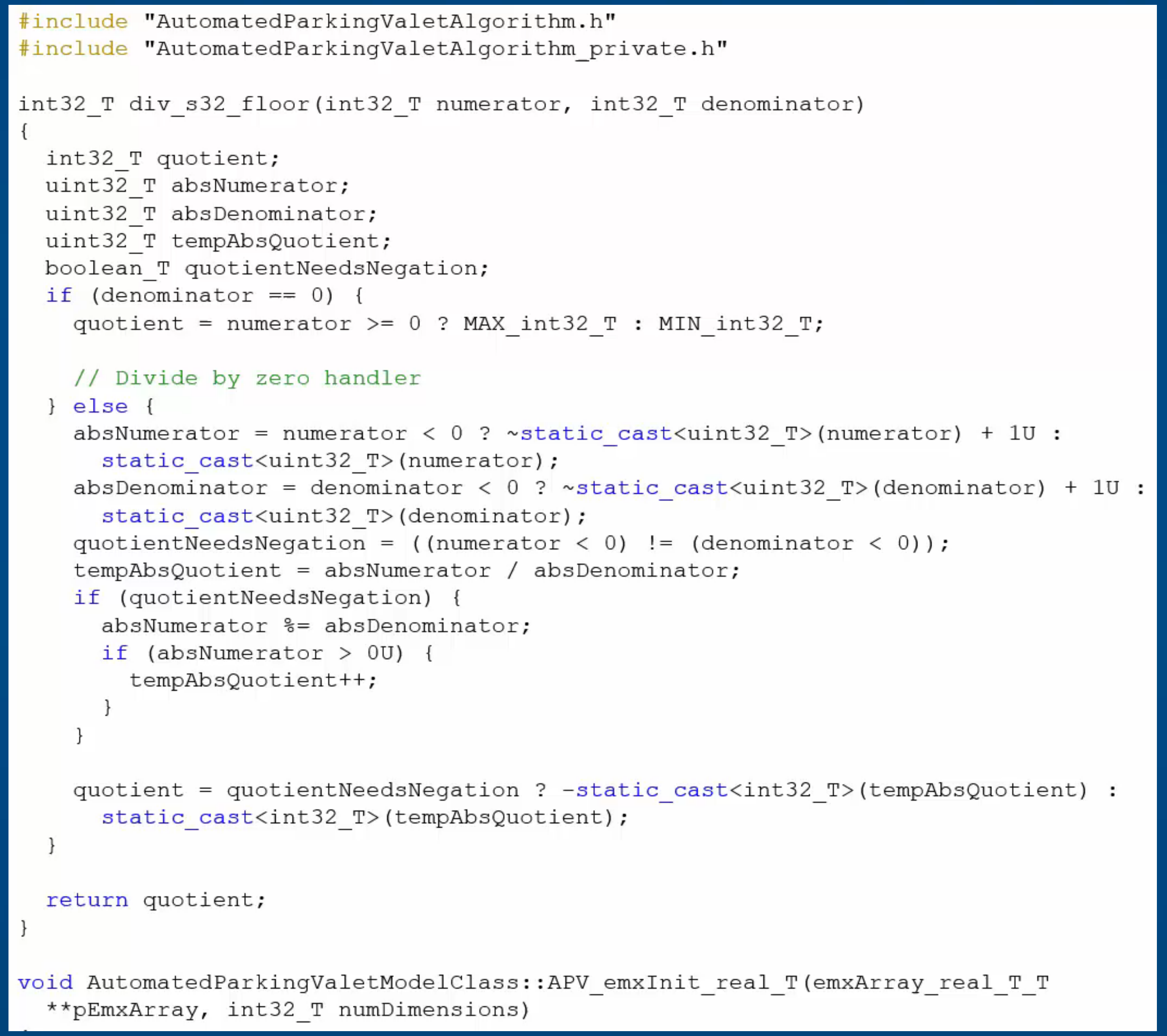
Train DQN Agent for Lane Keeping Assist

Reinforcement Learning Toolbox

AutoVrtlEnv (64-bit, PCD3D_SM5)
WTHFUN

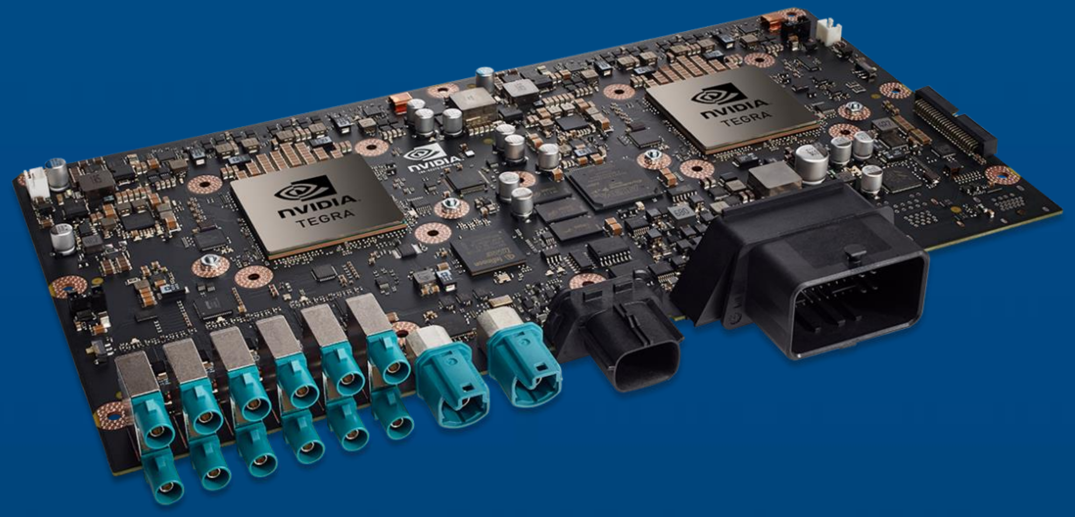

Modeling
Simulation
Automation
Coding

Coding

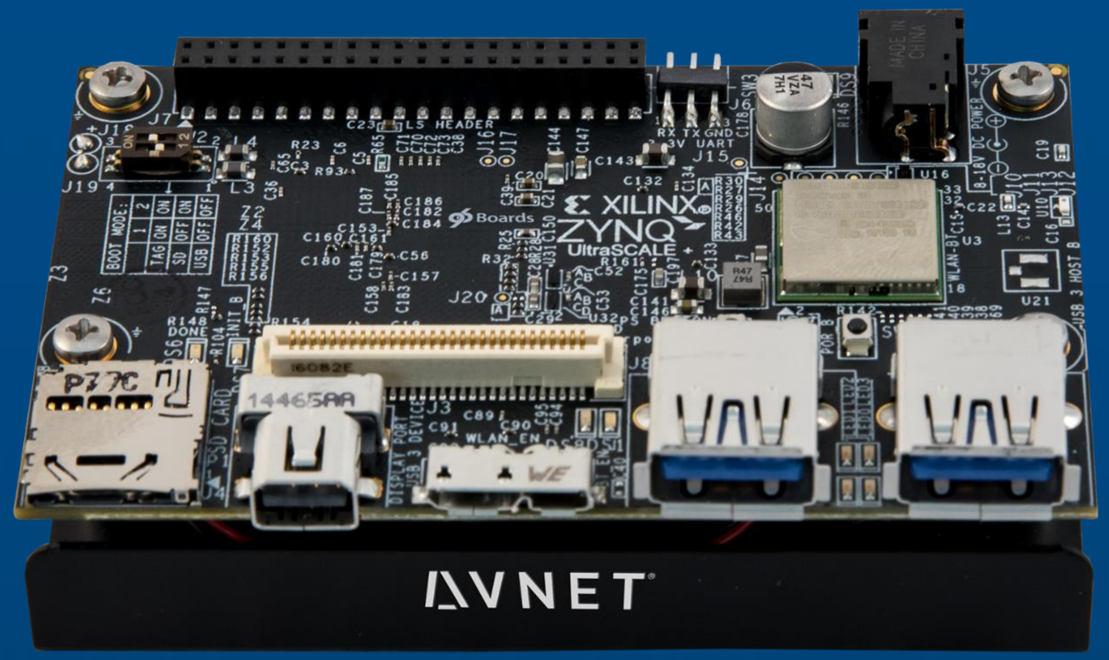

```
#include "AutomatedParkingValetAlgorithm.h"
#include "AutomatedParkingValetAlgorithm_private.h"

int32_T div_s32_floor(int32_T numerator, int32_T denominator)
{
  int32_T quotient;
  uint32_T absNumerator;
  uint32_T absDenominator;
  uint32_T tempAbsQuotient;
  boolean_T quotientNeedsNegation;
  if (denominator == 0) {
    quotient = numerator >= 0 ? MAX_int32_T : MIN_int32_T;

    // Divide by zero handler
  } else {
    absNumerator = numerator < 0 ? ~static_cast<uint32_T>(numerator) + 1U :
      static_cast<uint32_T>(numerator);
    absDenominator = denominator < 0 ? ~static_cast<uint32_T>(denominator) + 1U :
      static_cast<uint32_T>(denominator);
    quotientNeedsNegation = ((numerator < 0) != (denominator < 0));
    tempAbsQuotient = absNumerator / absDenominator;
    if (quotientNeedsNegation) {
      absNumerator %= absDenominator;
      if (absNumerator > 0U) {
        tempAbsQuotient++;
      }
    }

    quotient = quotientNeedsNegation ? -static_cast<int32_T>(tempAbsQuotient) :
      static_cast<int32_T>(tempAbsQuotient);
  }

  return quotient;
}

void AutomatedParkingValetModelClass::APV_emxInit_real_T(emxArray_real_T_T
  **pEmxArray, int32_T numDimensions)
```

Solutions for Vision and Deep Learning

GPU
Fastest



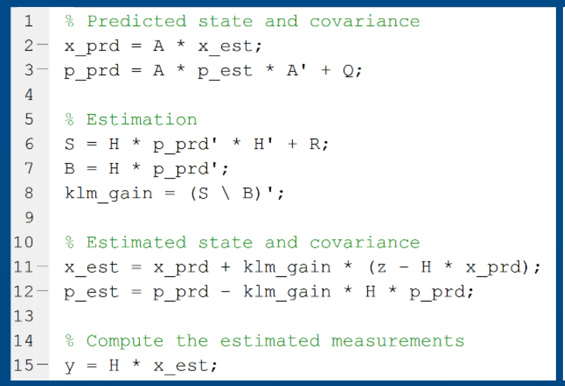
FPGA / ASIC
Lowest Power



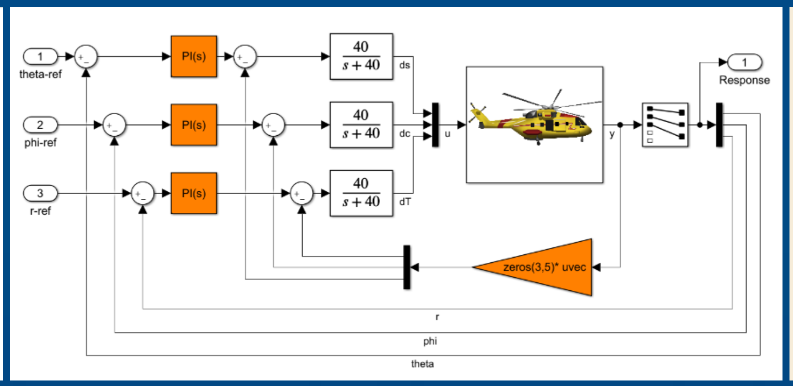
CPU
Low Cost

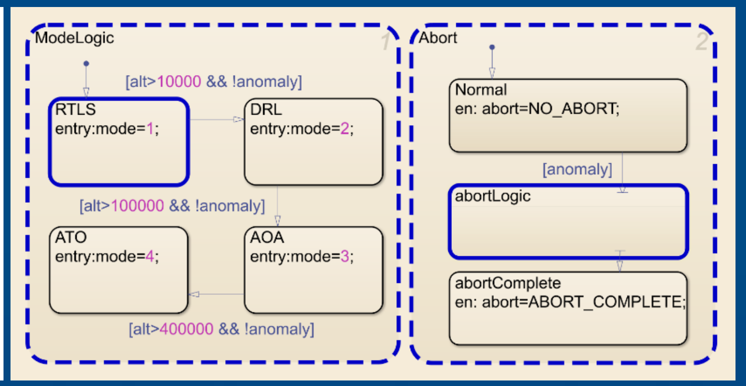
Model-Based Design

C/C++

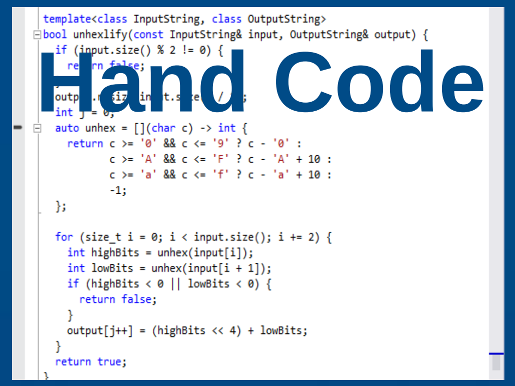


```
% Predicted state and covariance
x_prd = A * x_est;
p_prd = A * p_est * A' + Q;

% Estimation
S = H * p_prd' * H' + R;
B = H * p_prd';
klm_gain = (S \ B)';

% Estimated state and covariance
x_est = x_prd + klm_gain * (z - H * x_prd);
p_est = p_prd - klm_gain * H * p_prd;

% Compute the estimated measurements
y = H * x_est;
```





VS



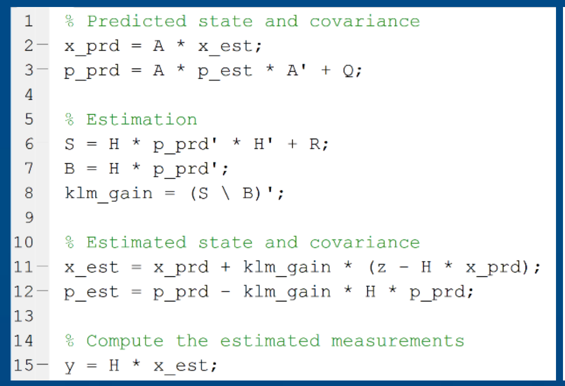
Hand Code

```
template<class InputString, class OutputString>
bool unhexlify(const InputString& input, OutputString& output) {
  if (input.size() % 2 != 0) {
    return false;
  }
  output.resize(input.size() / 2);
  int j = 0;
  auto unhex = [](char c) -> int {
    return c >= '0' && c <= '9' ? c - '0' :
           c >= 'A' && c <= 'F' ? c - 'A' + 10 :
           c >= 'a' && c <= 'f' ? c - 'a' + 10 :
           -1;
  };

  for (size_t i = 0; i < input.size(); i += 2) {
    int highBits = unhex(input[i]);
    int lowBits = unhex(input[i + 1]);
    if (highBits < 0 || lowBits < 0) {
      return false;
    }
    output[j++] = (highBits << 4) + lowBits;
  }
  return true;
}
```

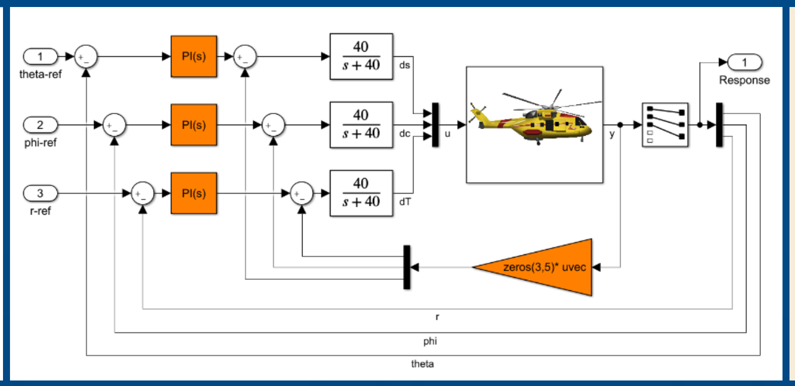
- High level of abstraction
- Advanced analysis tools
- Automatic code generation

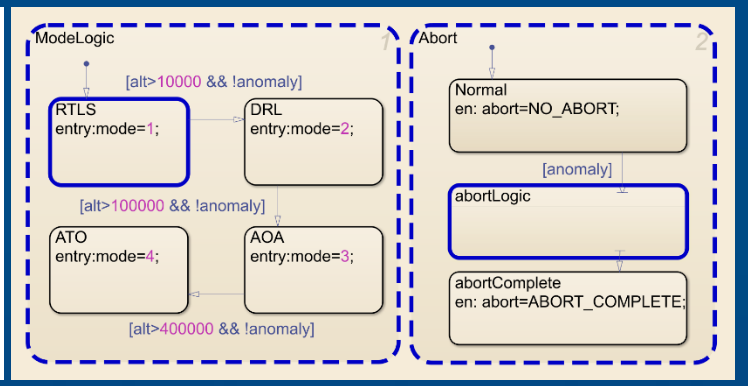
Model-Based Design

C/C++ Libraries



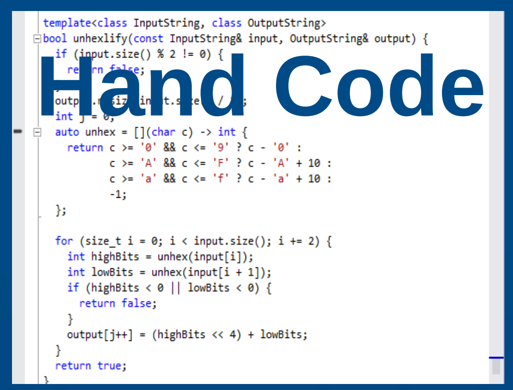
```
% Predicted state and covariance
x_prd = A * x_est;
p_prd = A * p_est * A' + Q;

% Estimation
S = H * p_prd' * H' + R;
B = H * p_prd';
klm_gain = (S \ B)';

% Estimated state and covariance
x_est = x_prd + klm_gain * (z - H * x_prd);
p_est = p_prd - klm_gain * H * p_prd;

% Compute the estimated measurements
y = H * x_est;
```





+

Hand Code



```
template<class InputString, class OutputString>
bool unhexlify(const InputString& input, OutputString& output) {
  if (input.size() % 2 != 0) {
    return false;
  }
  output.resize(input.size() / 2);
  int j = 0;
  auto unhex = [](char c) -> int {
    return c >= '0' && c <= '9' ? c - '0' :
           c >= 'A' && c <= 'F' ? c - 'A' + 10 :
           c >= 'a' && c <= 'f' ? c - 'a' + 10 :
           -1;
  };

  for (size_t i = 0; i < input.size(); i += 2) {
    int highBits = unhex(input[i]);
    int lowBits = unhex(input[i + 1]);
    if (highBits < 0 || lowBits < 0) {
      return false;
    }
    output[j++] = (highBits << 4) + lowBits;
  }
  return true;
}
```

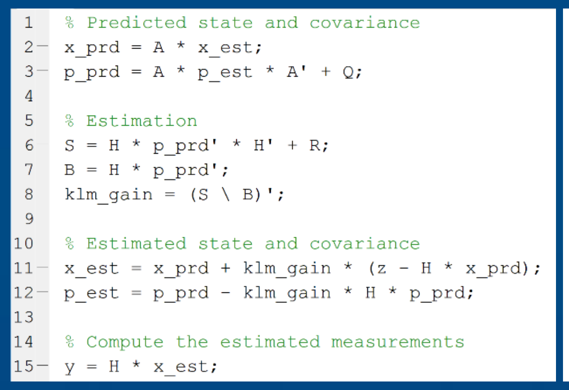
Internal Libraries

Vendor Libraries

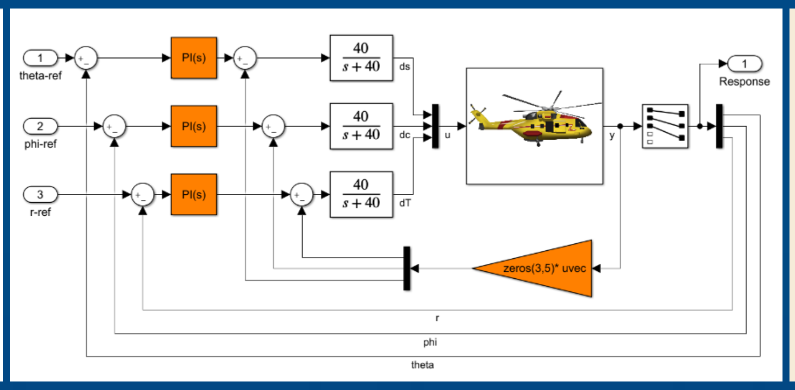
- No wrappers
- No data typing
- No data copies

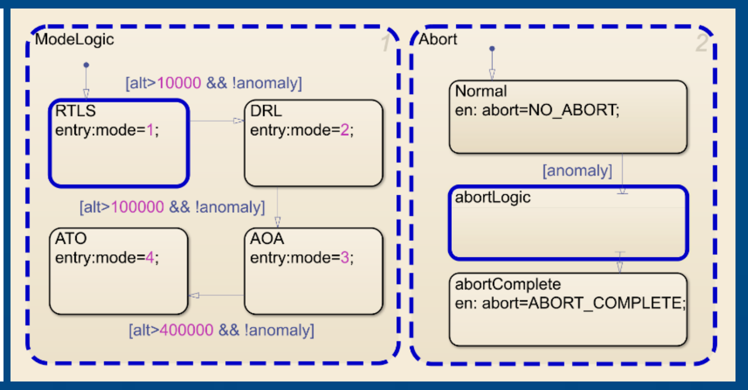
Model-Based Design

C/C++ Libraries



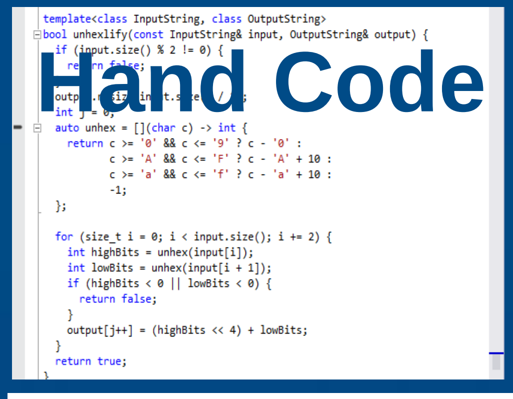
```
% Predicted state and covariance
x_prd = A * x_est;
p_prd = A * p_est * A' + Q;

% Estimation
S = H * p_prd' * H' + R;
B = H * p_prd';
klm_gain = (S \ B)';

% Estimated state and covariance
x_est = x_prd + klm_gain * (z - H * x_prd);
p_est = p_prd - klm_gain * H * p_prd;

% Compute the estimated measurements
y = H * x_est;
```





- No wrappers
- No data typing
- No data copies

Hand Code



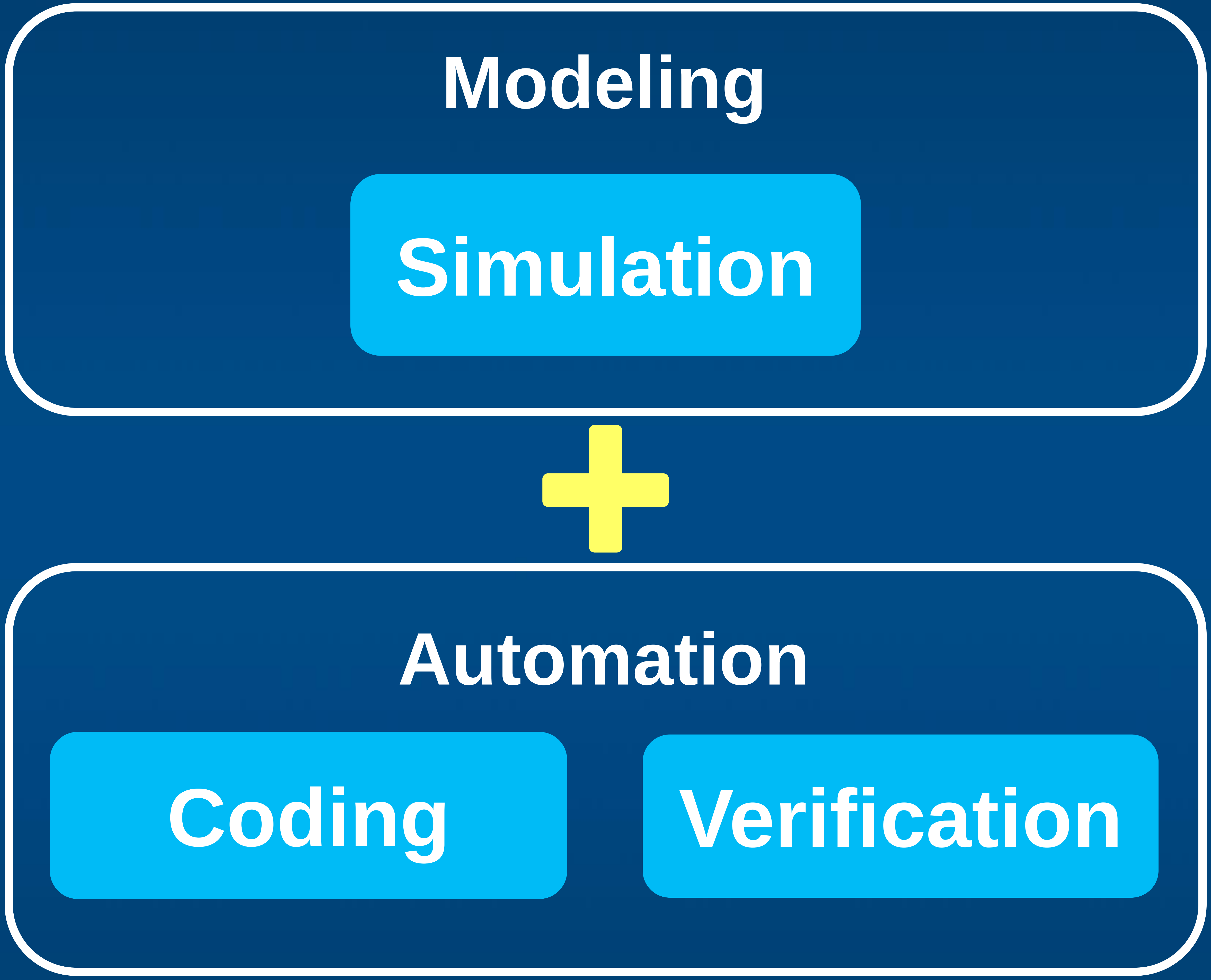
```
template<class InputString, class OutputString>
bool unhexlify(const InputString& input, OutputString& output) {
  if (input.size() % 2 != 0) {
    return false;
  }
  output.resize(input.size() / 2);
  int j = 0;
  auto unhex = [](char c) -> int {
    return c >= '0' && c <= '9' ? c - '0' :
           c >= 'A' && c <= 'F' ? c - 'A' + 10 :
           c >= 'a' && c <= 'f' ? c - 'a' + 10 :
           -1;
  };

  for (size_t i = 0; i < input.size(); i += 2) {
    int highBits = unhex(input[i]);
    int lowBits = unhex(input[i + 1]);
    if (highBits < 0 || lowBits < 0) {
      return false;
    }
    output[j++] = (highBits << 4) + lowBits;
  }
  return true;
}
```

Internal Libraries

Vendor Libraries

Middleware

ROS AUTOSAR DDS



Modeling

Simulation

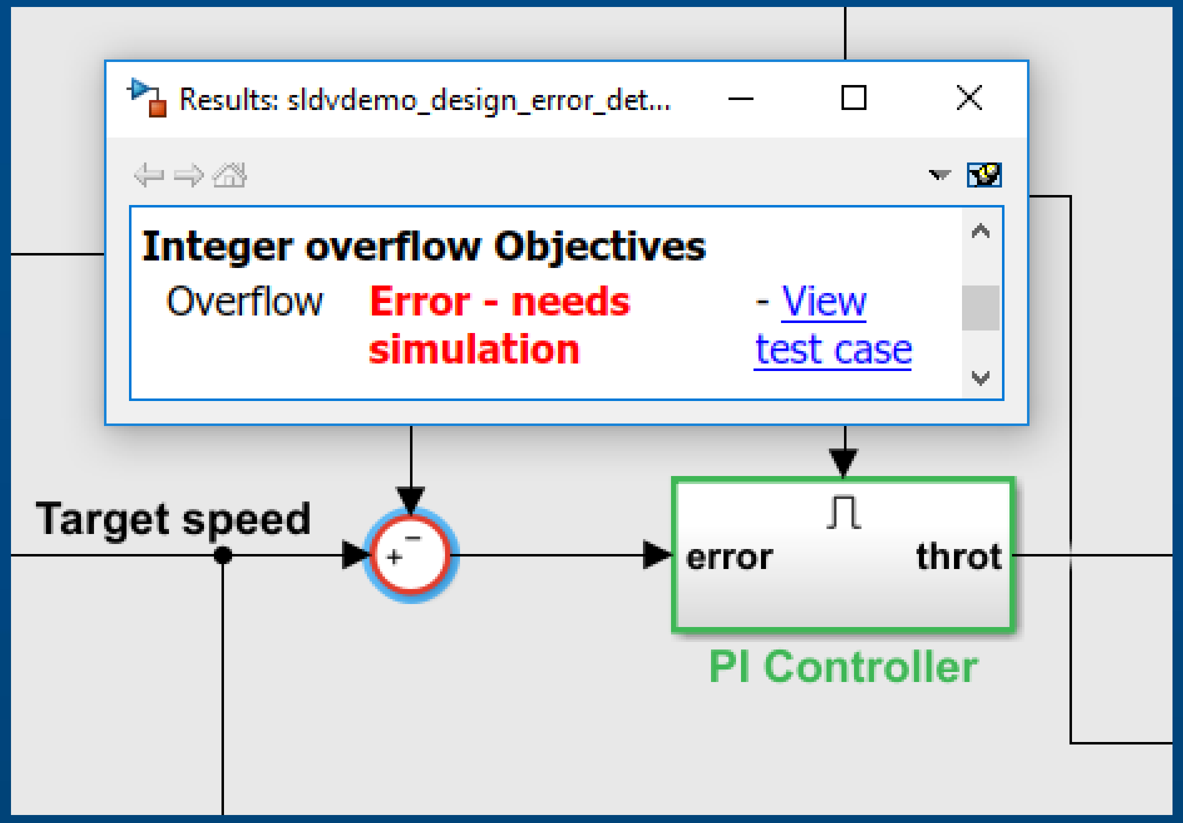
Automation

Coding

Verification

Automated Test and Verification

Find bugs



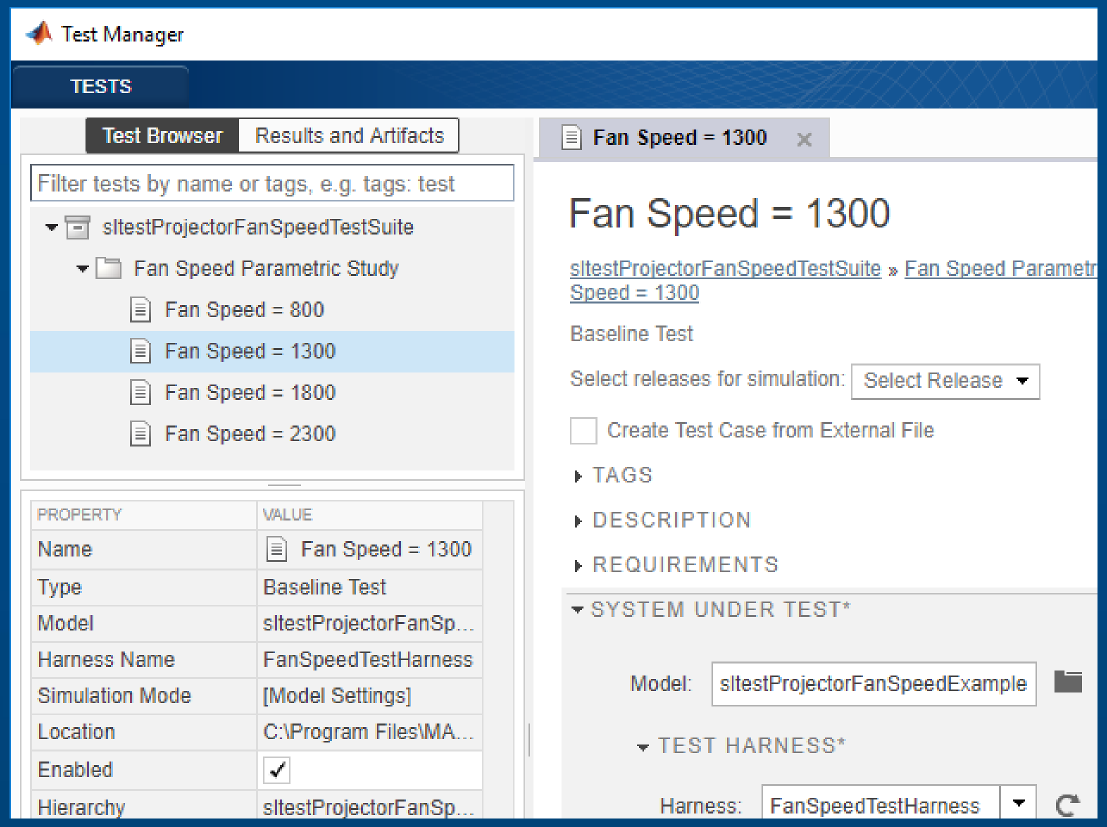
Results: sldvdemo_design_error_det...

Integer overflow Objectives

Overflow **Error - needs simulation** - View test case

Simulink Design Verifier
Polyspace Bug Finder

Manage tests



Test Manager

TESTS

Test Browser | Results and Artifacts

Filter tests by name or tags, e.g. tags: test

- sltestProjectorFanSpeedTestSuite
  - Fan Speed Parametric Study
    - Fan Speed = 800
    - Fan Speed = 1300
    - Fan Speed = 1800
    - Fan Speed = 2300

PROPERTY	VALUE
Name	Fan Speed = 1300
Type	Baseline Test
Model	sltestProjectorFanSp...
Harness Name	FanSpeedTestHarness
Simulation Mode	[Model Settings]
Location	C:\Program Files\MA...
Enabled	✓
Hierarchy	sltestProjectorFanSp...

Fan Speed = 1300

Fan Speed = 1300

sltestProjectorFanSpeedTestSuite » Fan Speed Parametr Speed = 1300

Baseline Test

Select releases for simulation: Select Release

□ Create Test Case from External File

TAGS

DESCRIPTION

REQUIREMENTS

SYSTEM UNDER TEST*

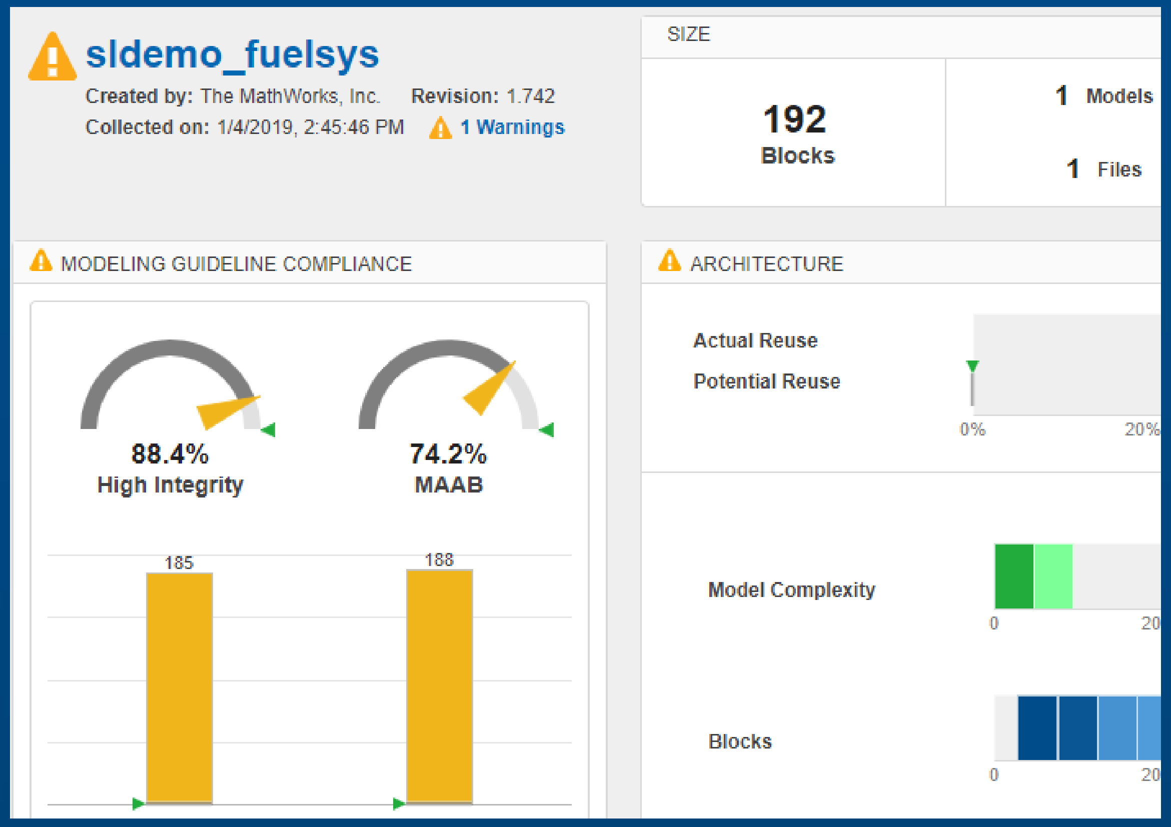
Model: sltestProjectorFanSpeedExample

TEST HARNESS*

Harness: FanSpeedTestHarness

Simulink Test

Check & Coverage



sldemo_fuelsys

Created by: The MathWorks, Inc. Revision: 1.742

Collected on: 1/4/2019, 2:45:46 PM 1 Warnings

SIZE

192 Blocks

1 Models

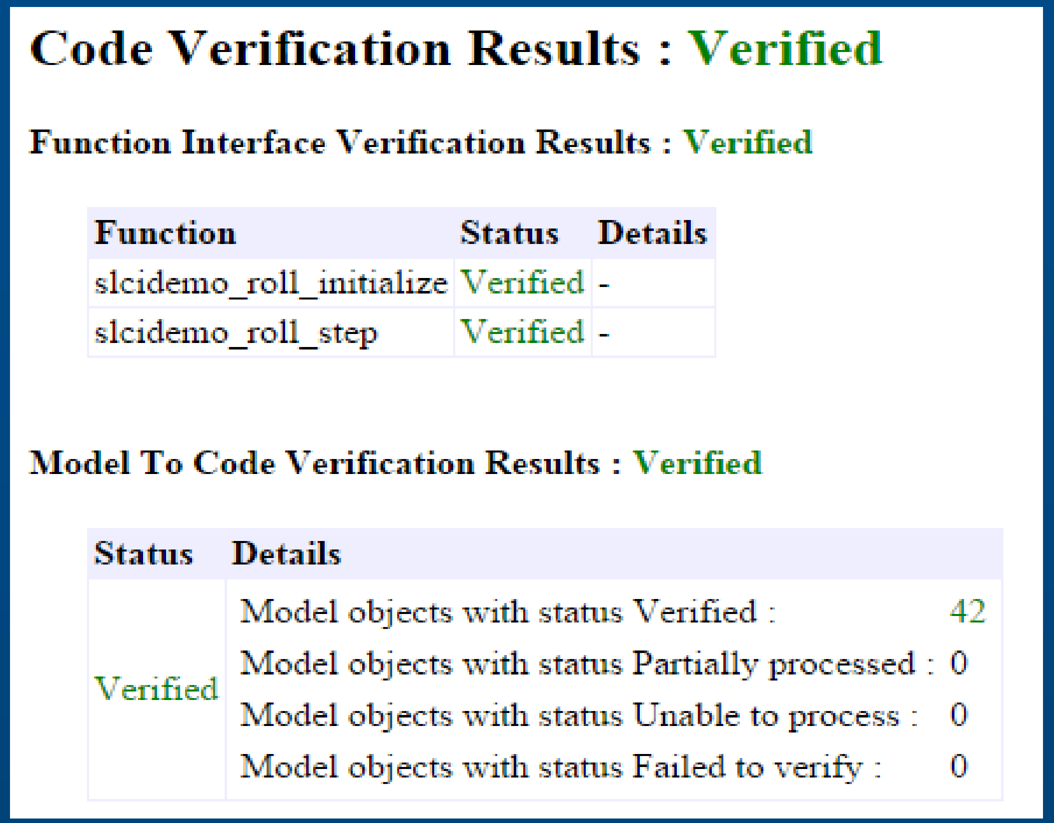
1 Files

MODELING GUIDELINE COMPLIANCE

ARCHITECTURE

Simulink Check
Simuink Coverage

Inspect code



Code Verification Results : Verified

Function Interface Verification Results : Verified

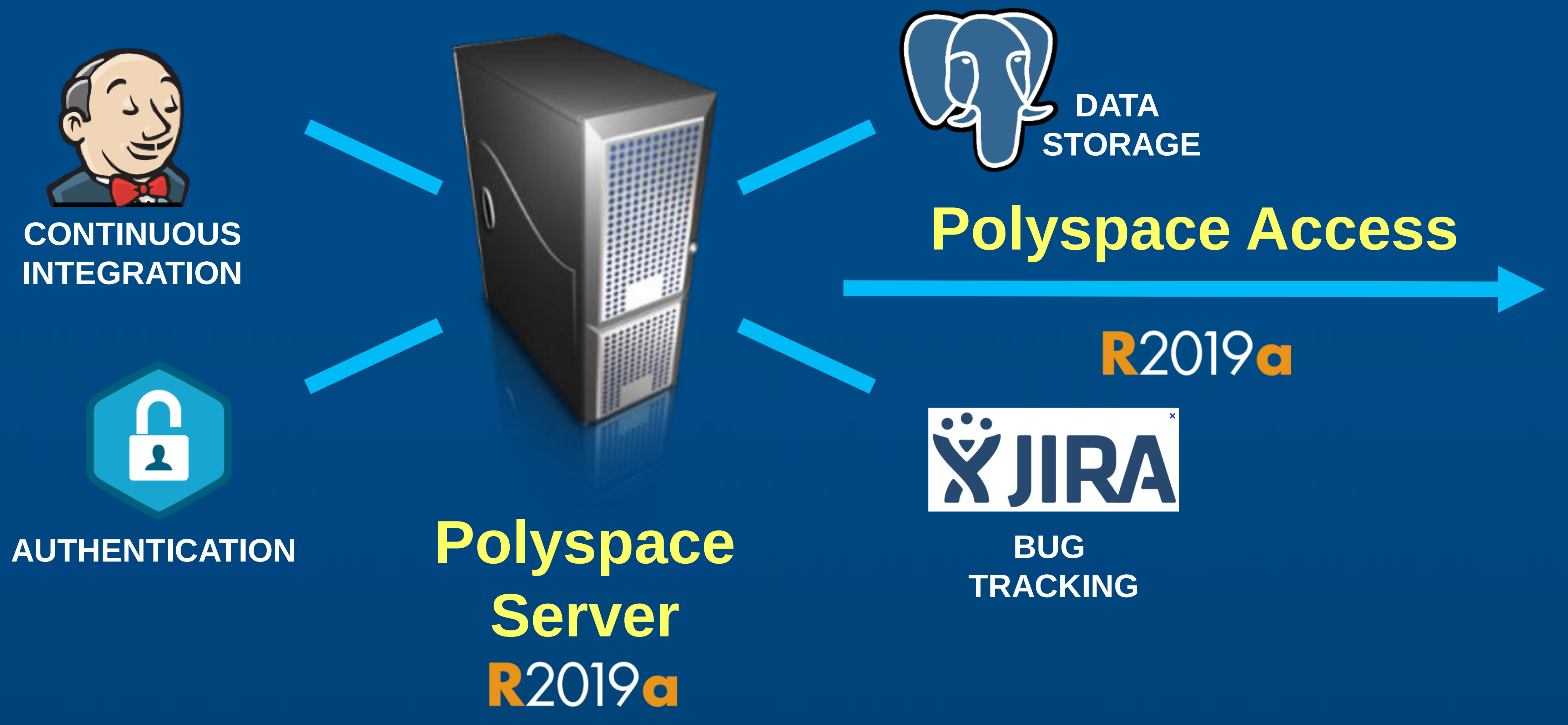
Function	Status	Details
slcidemo_roll_initialize	Verified	-
slcidemo_roll_step	Verified	-

Model To Code Verification Results : Verified

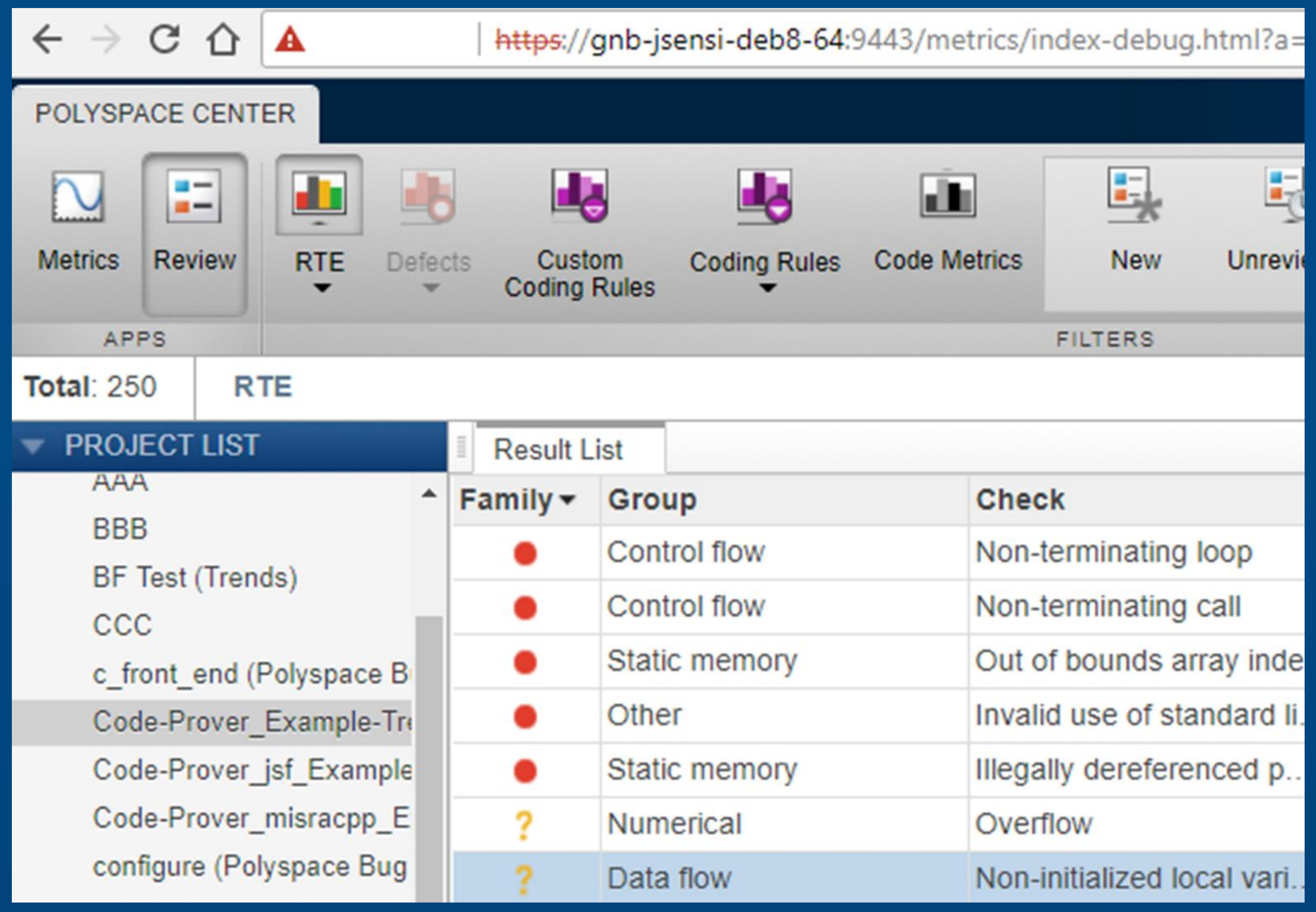
Status	Details	
Verified	Model objects with status Verified :	42
	Model objects with status Partially processed :	0
	Model objects with status Unable to process :	0
	Model objects with status Failed to verify :	0

Simulink Code Inspector

Online Access for Test and Verification



Web browser



Total: 250 RTE

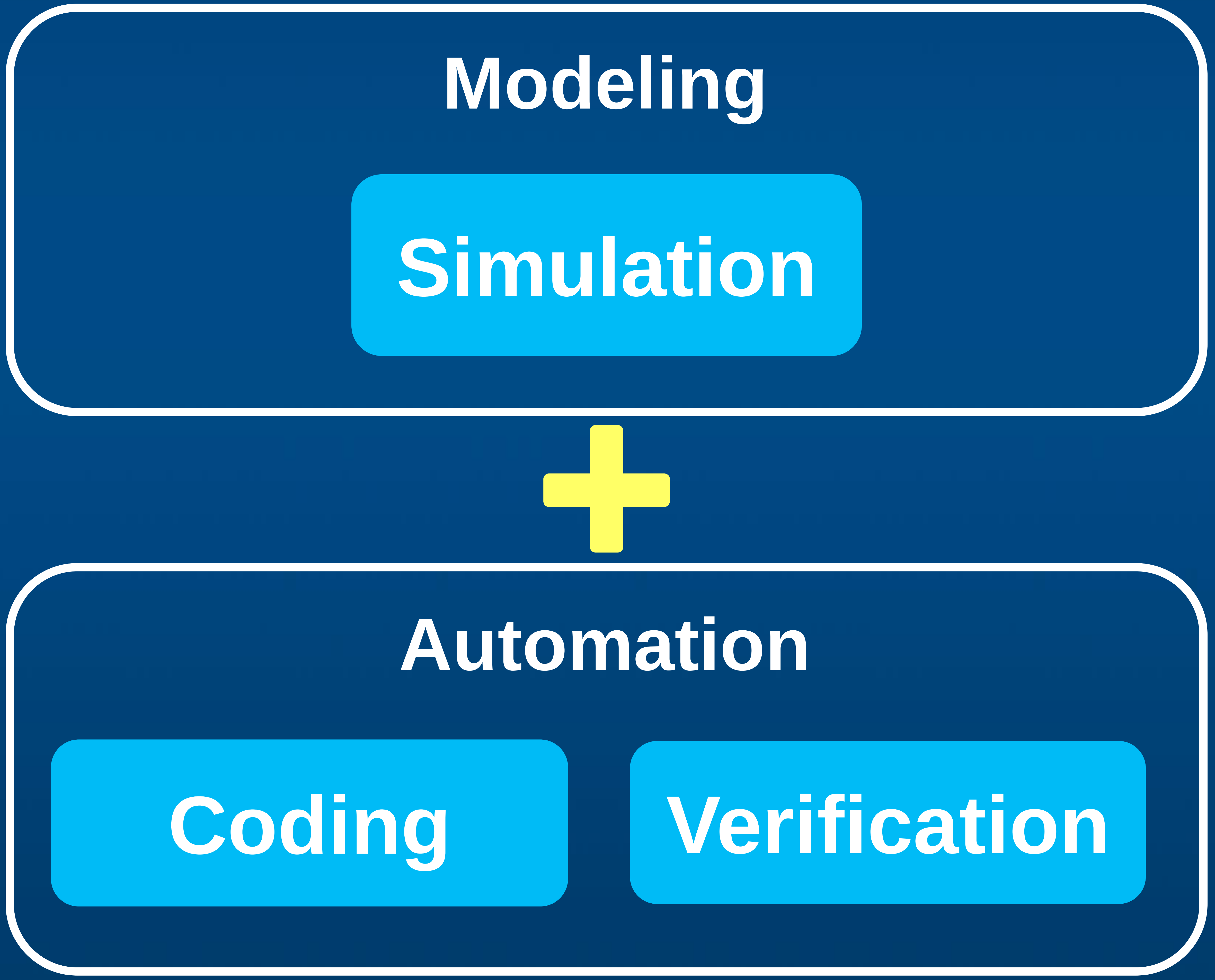
PROJECT LIST
AAA
BBB
BF Test (Trends)
CCC
c_front_end (Polyspace B
Code-Prover_Example-Tr
Code-Prover_jsf_Example
Code-Prover_misracpp_E
configure (Polyspace Bug

Result List

Family	Group	Check
●	Control flow	Non-terminating loop
●	Control flow	Non-terminating call
●	Static memory	Out of bounds array inde
●	Other	Invalid use of standard li..
●	Static memory	Illegally dereferenced p..
?	Numerical	Overflow
?	Data flow	Non-initialized local vari..

Model-Based Design

Systematic use of models throughout the development process



Modeling

Simulation

Automation

Coding

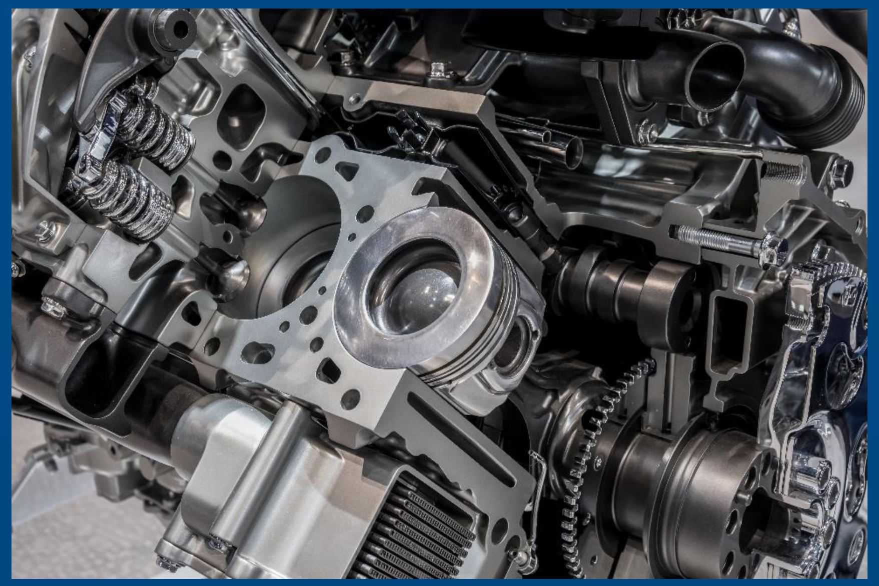
Verification

Fast repeatable tests

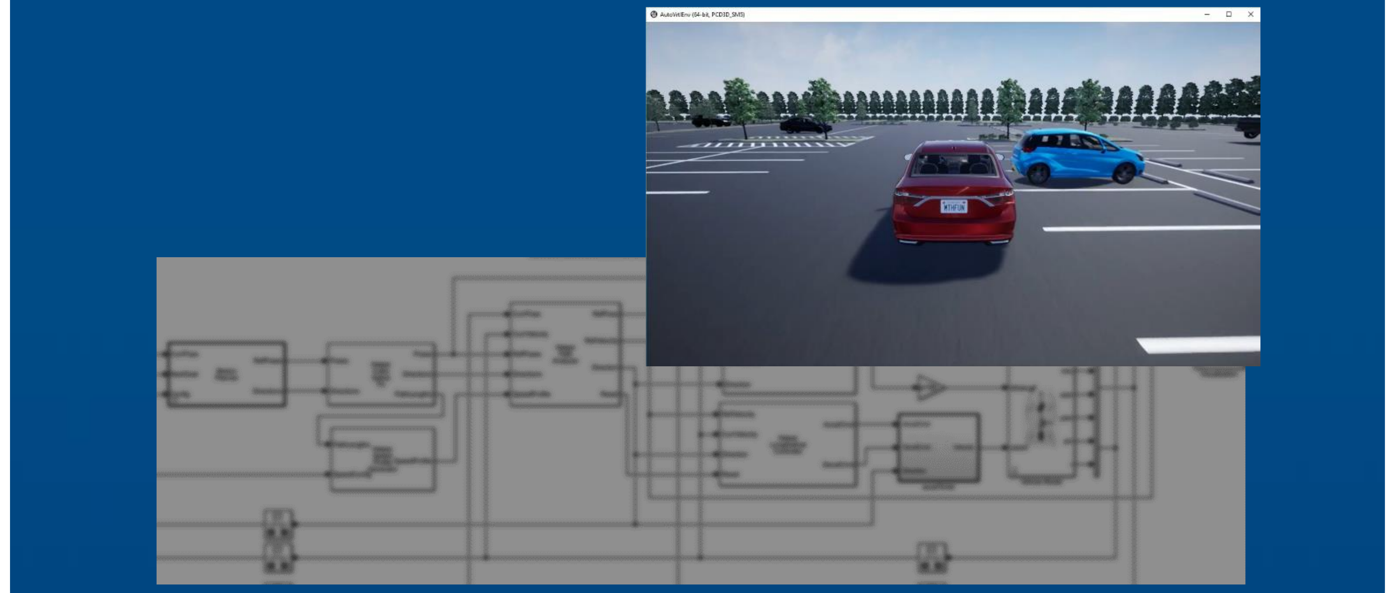
Fast agile development loops

Who will be successful in the future?

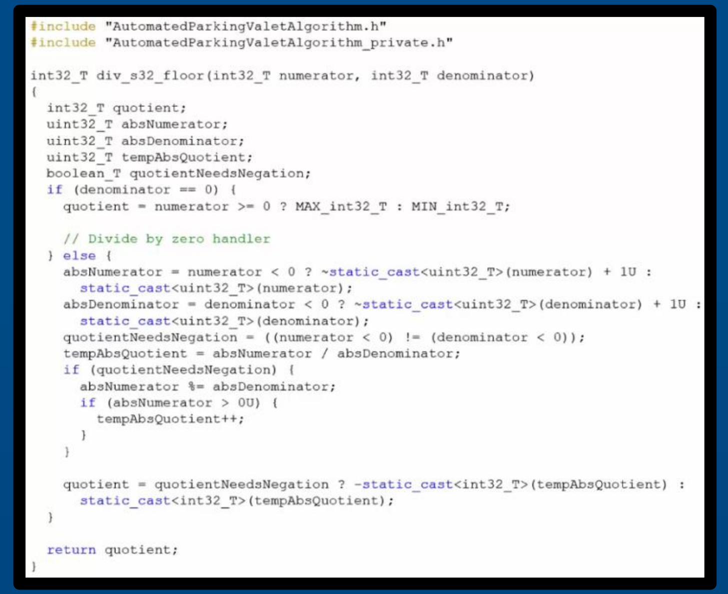
Mechanical-centric



Model-centric



Software-centric



```
#include "AutomatedParkingValetAlgorithm.h"
#include "AutomatedParkingValetAlgorithm_private.h"

int32_T div_s32_floor(int32_T numerator, int32_T denominator)
{
  int32_T quotient;
  uint32_T absNumerator;
  uint32_T absDenominator;
  uint32_T tempAbsQuotient;
  boolean_T quotientNeedsNegation;
  if (denominator == 0) {
    quotient = numerator >= 0 ? MAX_int32_T : MIN_int32_T;

    // Divide by zero handler
  } else {
    absNumerator = numerator < 0 ? ~static_cast<uint32_T>(numerator) + 1U :
      static_cast<uint32_T>(numerator);
    absDenominator = denominator < 0 ? ~static_cast<uint32_T>(denominator) + 1U :
      static_cast<uint32_T>(denominator);
    quotientNeedsNegation = ((numerator < 0) != (denominator < 0));
    tempAbsQuotient = absNumerator / absDenominator;
    if (quotientNeedsNegation) {
      absNumerator %= absDenominator;
      if (absNumerator > 0U) {
        tempAbsQuotient++;
      }
    }

    quotient = quotientNeedsNegation ? -static_cast<int32_T>(tempAbsQuotient) :
      static_cast<int32_T>(tempAbsQuotient);
  }

  return quotient;
}
```

Comprehensive models
Simulation based testing
Generate code and automate verification

Enjoy the conference