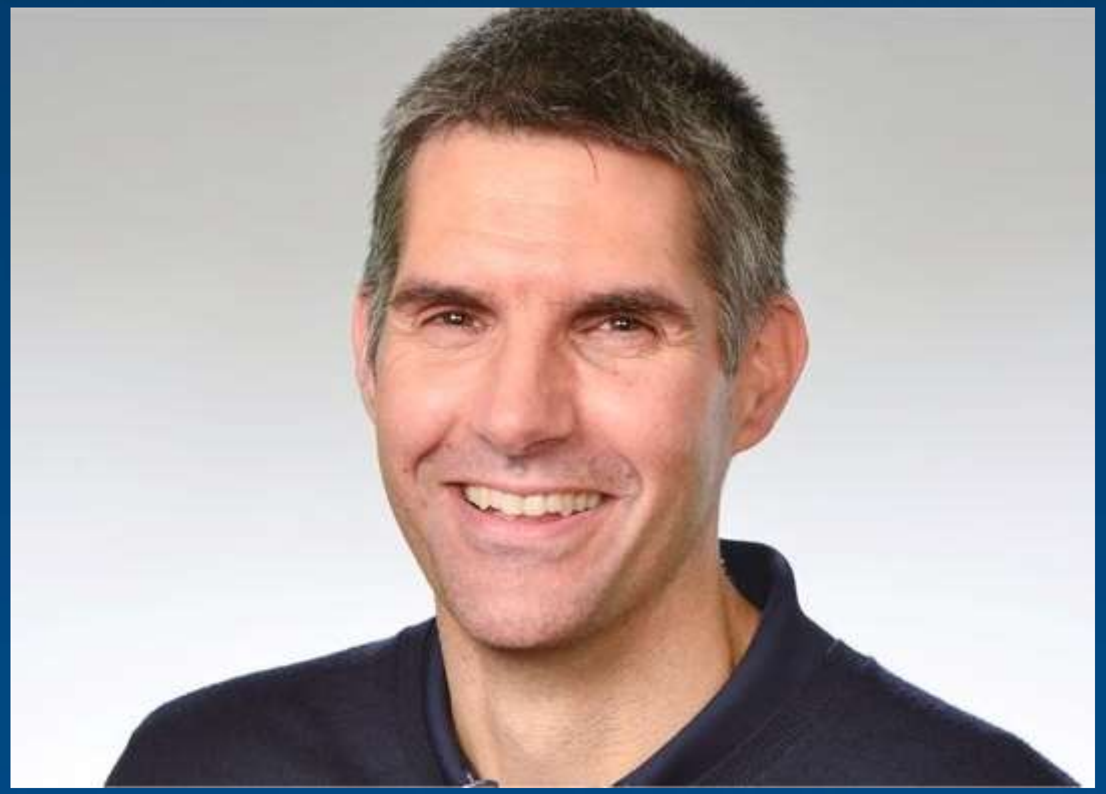
A look to the future with Model-Based Design



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+

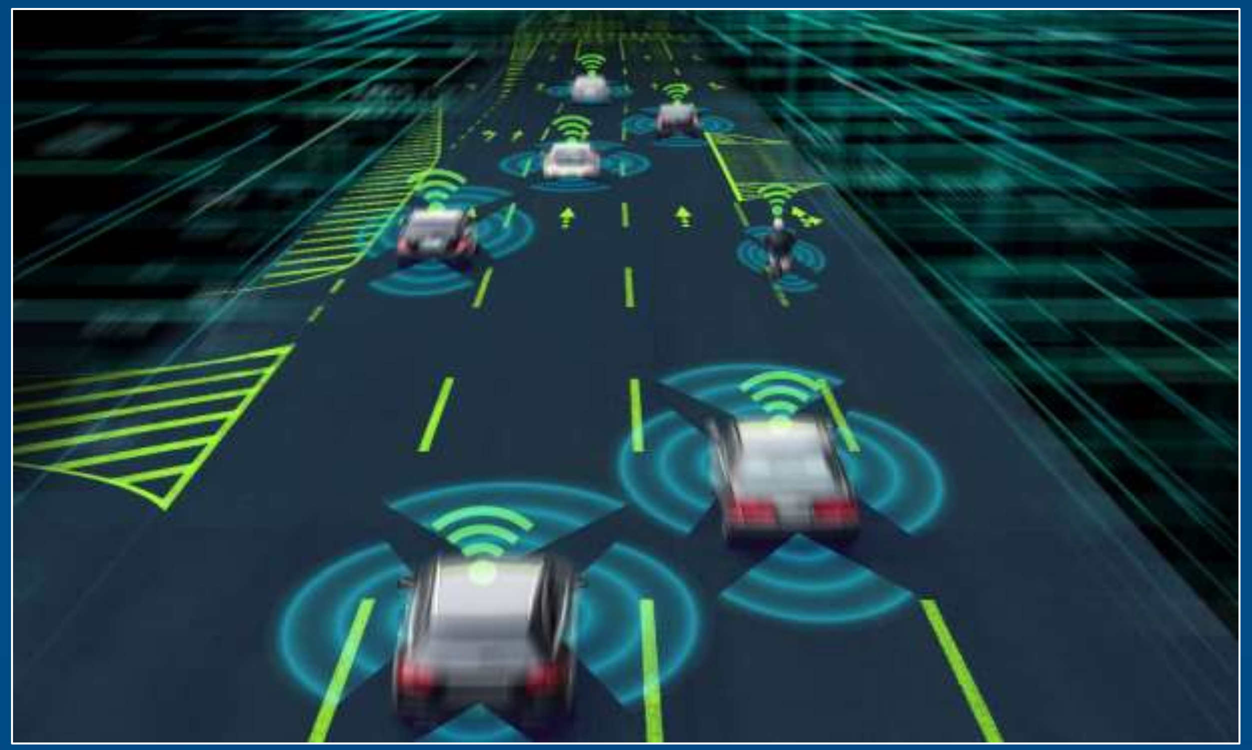
in 2018 revenues with 60% from outside the US

Privately held

and profitable every year

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

Marc Andreessen
Founder of Netscape,
Renowned Venture capitalist

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

Software is reshaping the automotive industry

Augmenting control with machine learning (BMW)

Trailer backup assist (Ford)

Autonomous driving (Voyage)

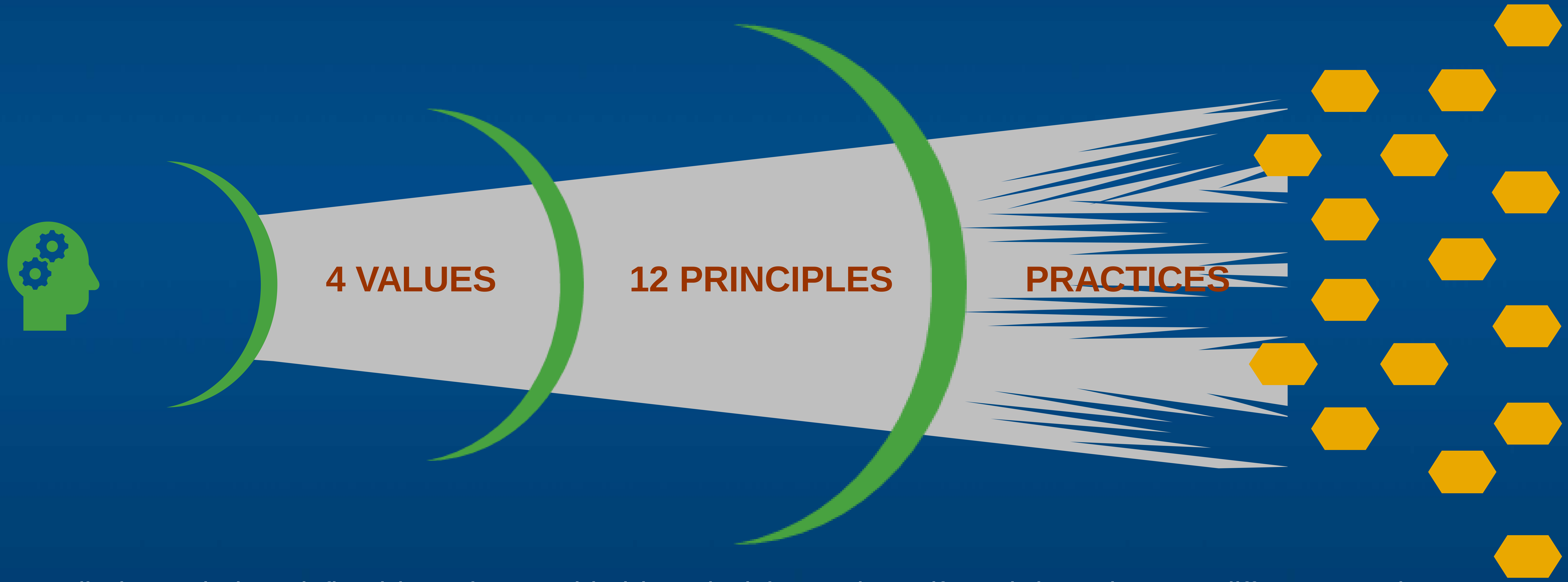
Agile Values

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

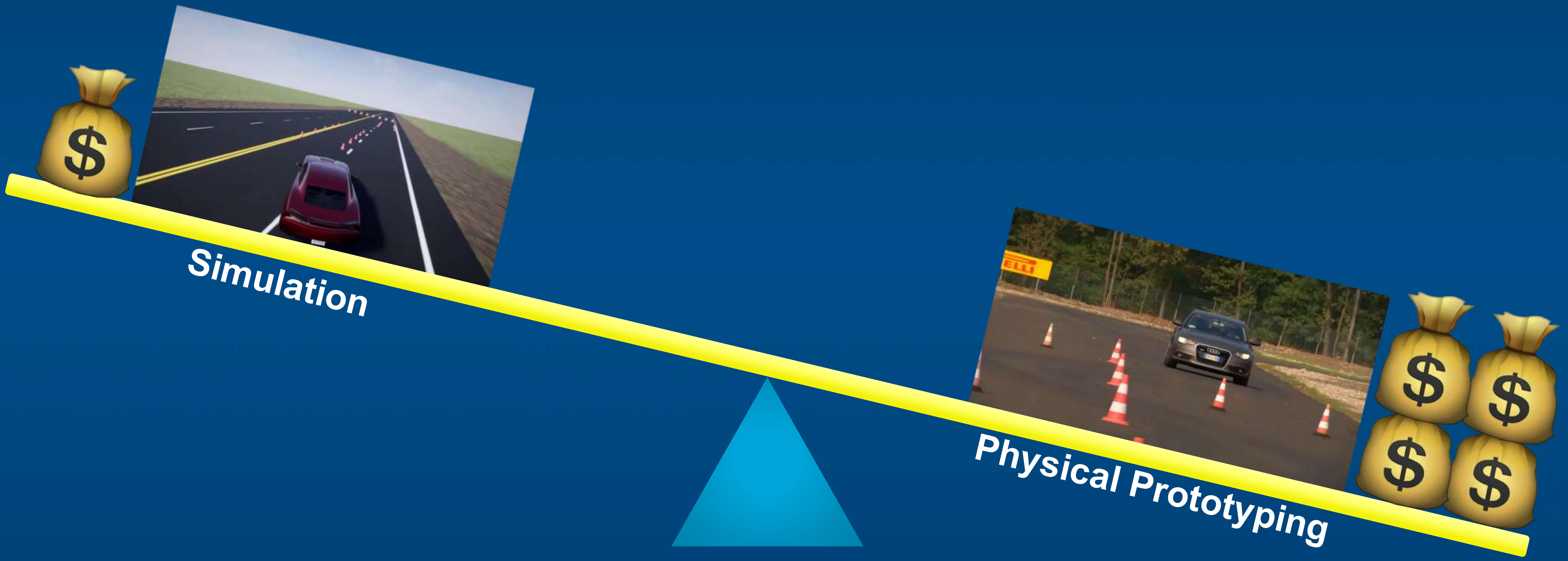


Agile is a mindset defined by values, guided by principles and manifested through many different practices. Agile practitioners select practices based on their needs.

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

Typical agile development workflow

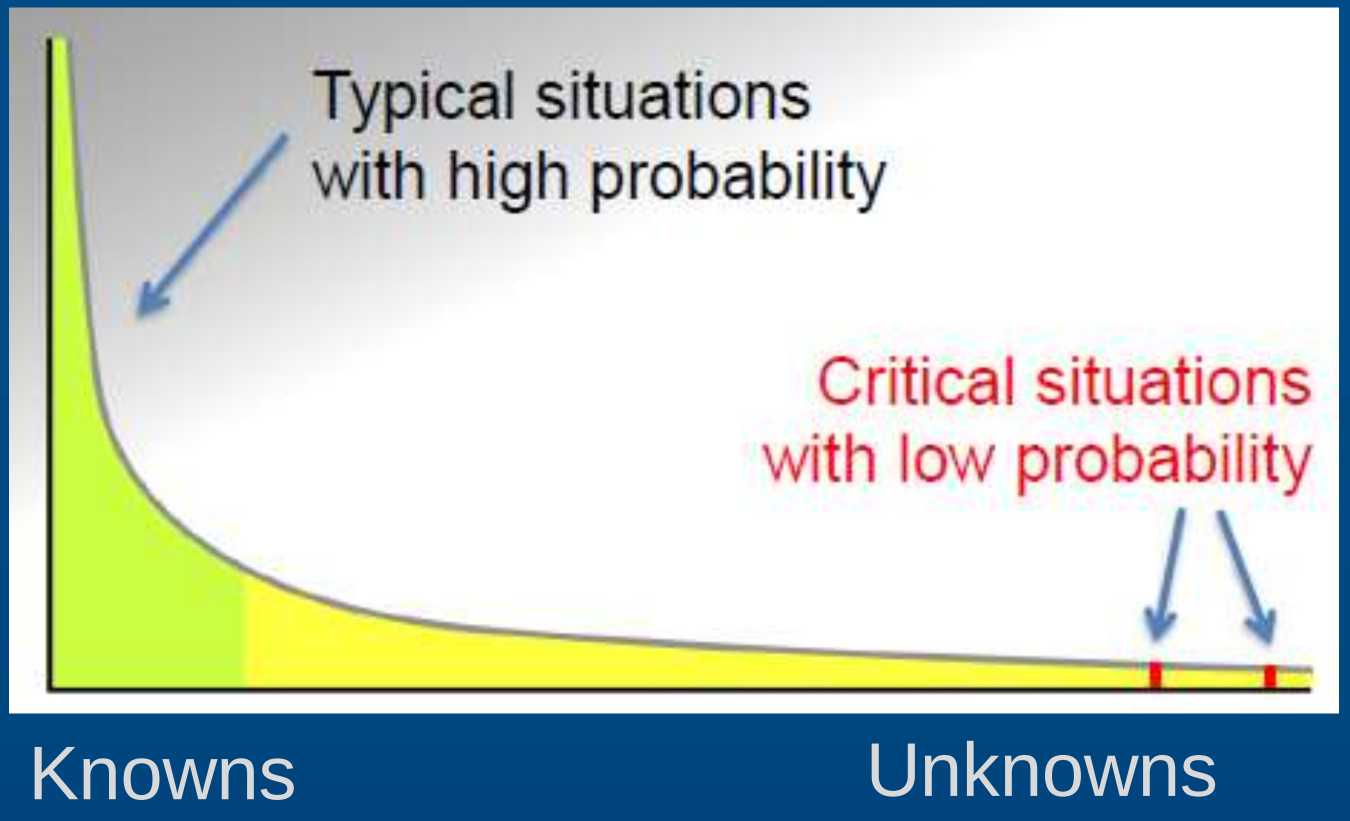
Models == Understanding


Simulation
Physical Prototyping

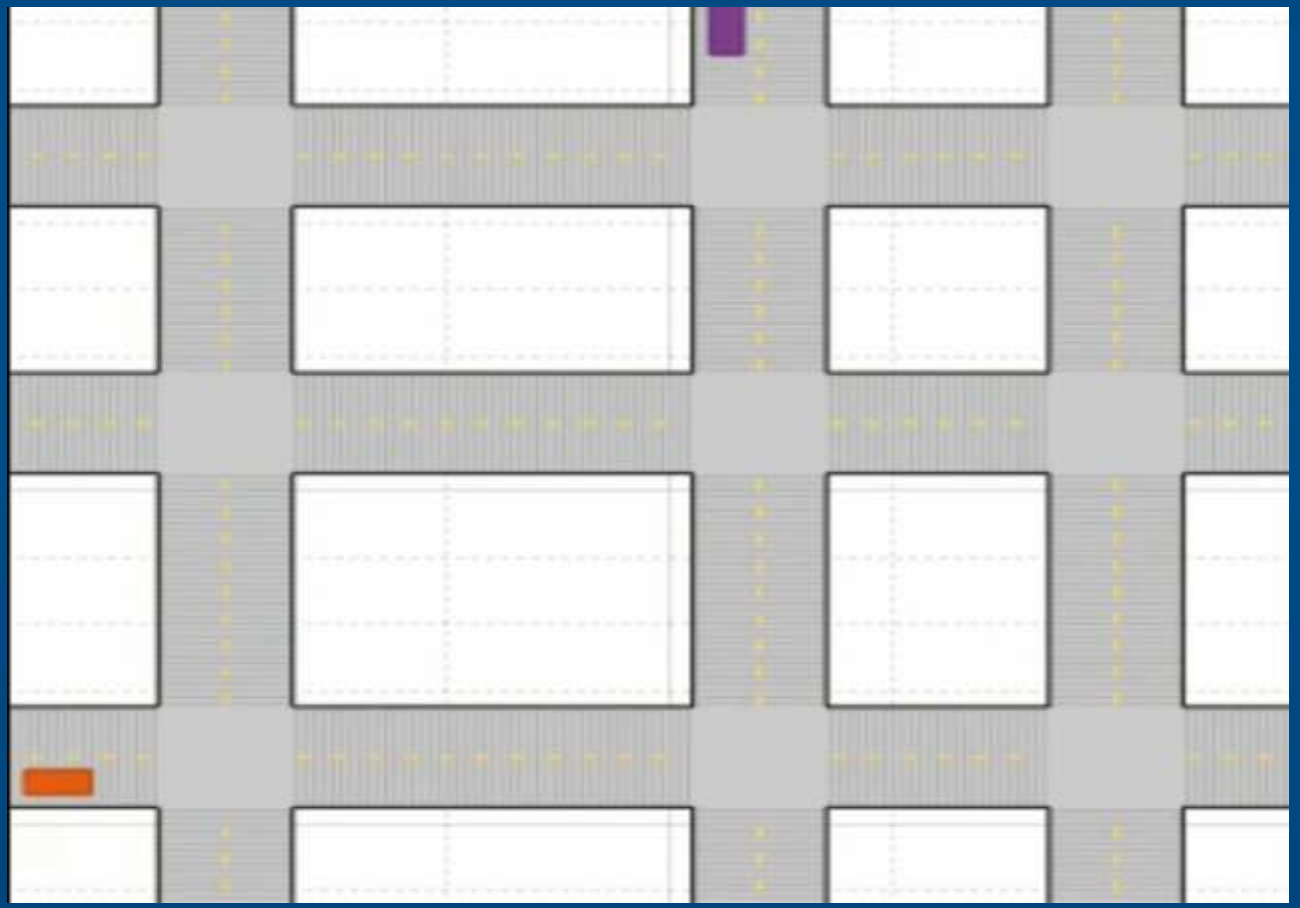
Simulation

Physical Prototyping

Simulation is key to Level 4-5 autonomy



Critical situations are in the long-tail*

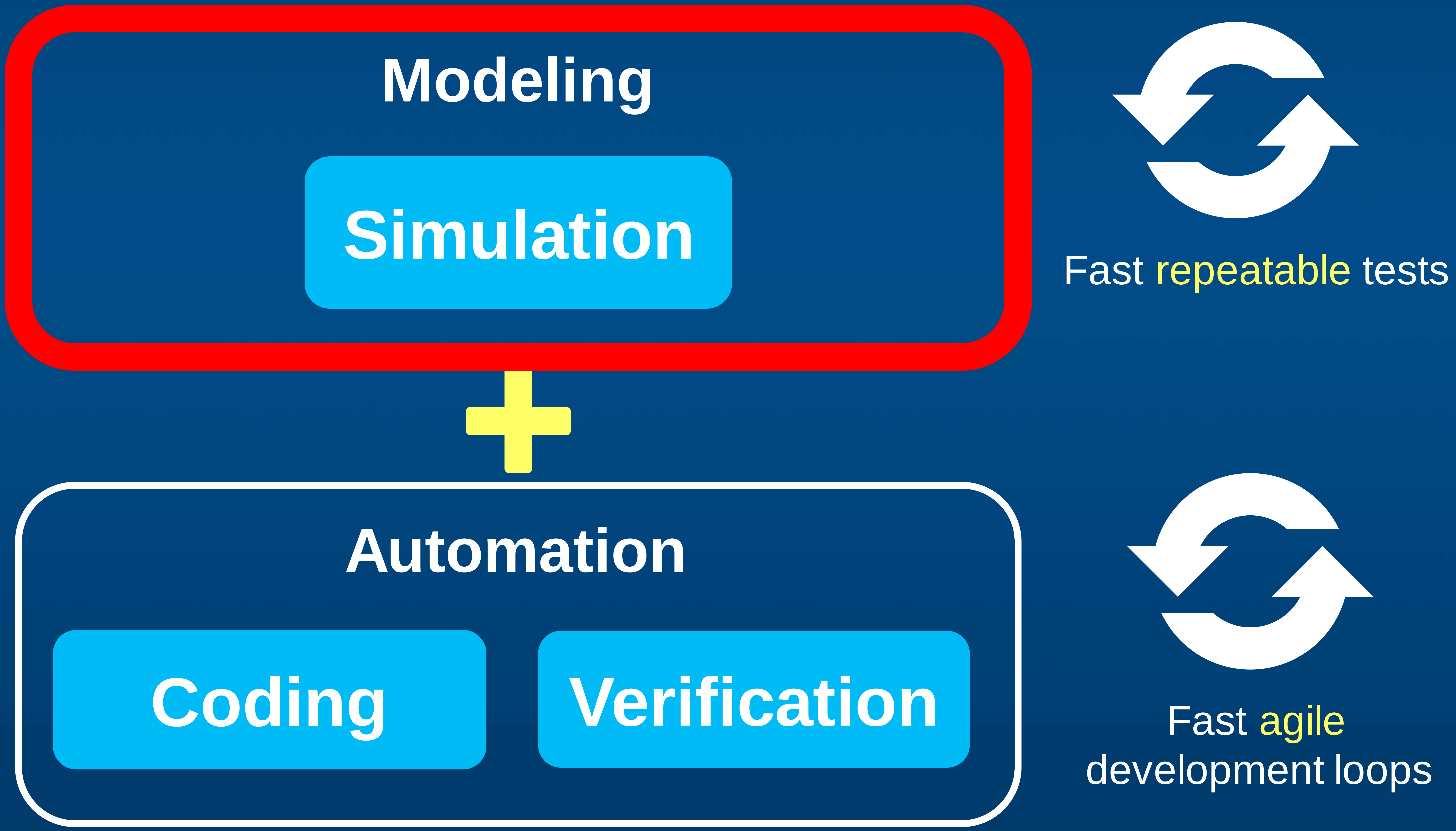


Simulation helps achieve this improbable task

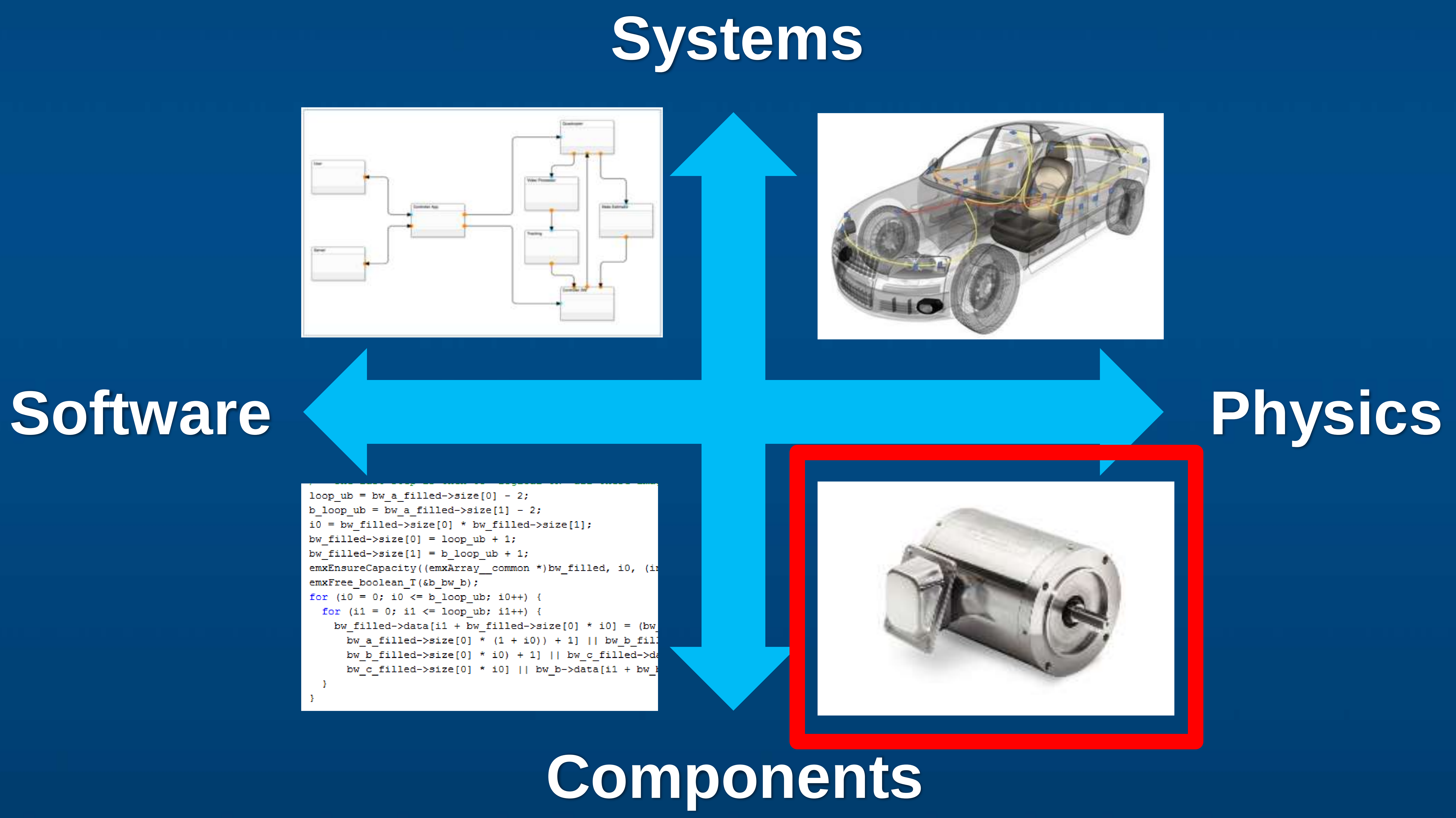
*Source: Center for Artificial Intelligence, Saarland University

Model-Based Design

Systematic use of models throughout the development process

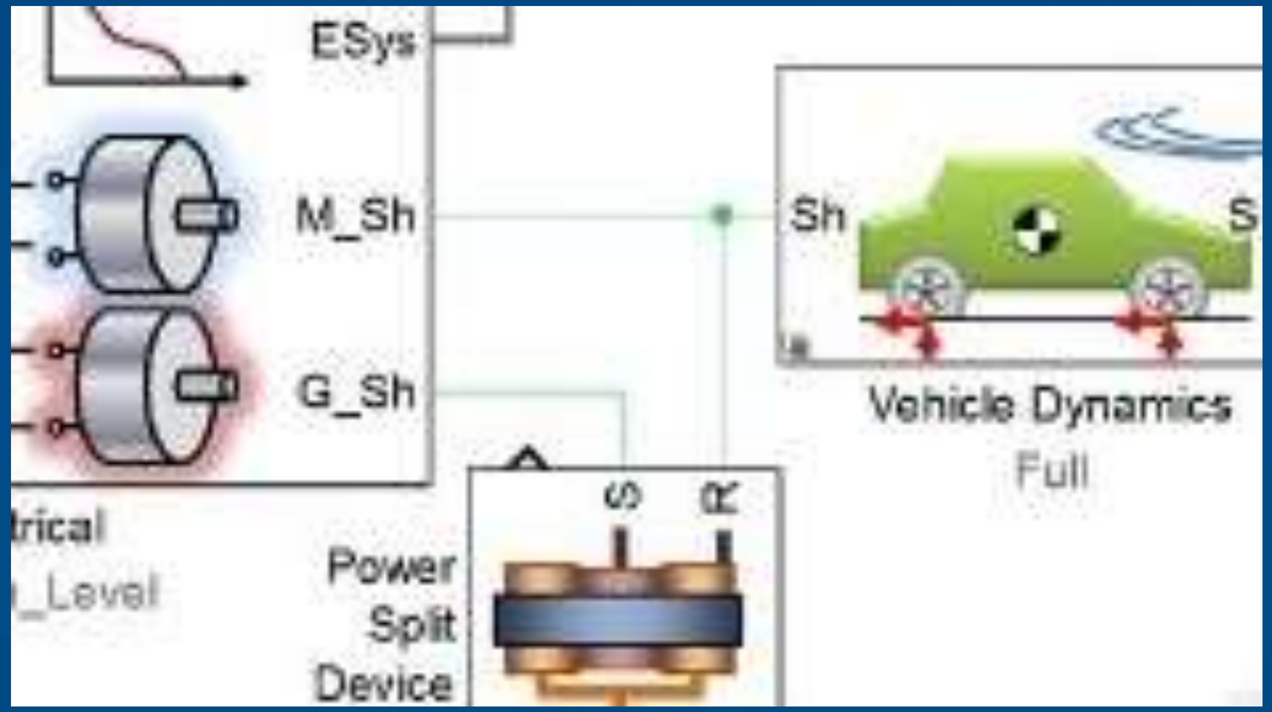


Types of models

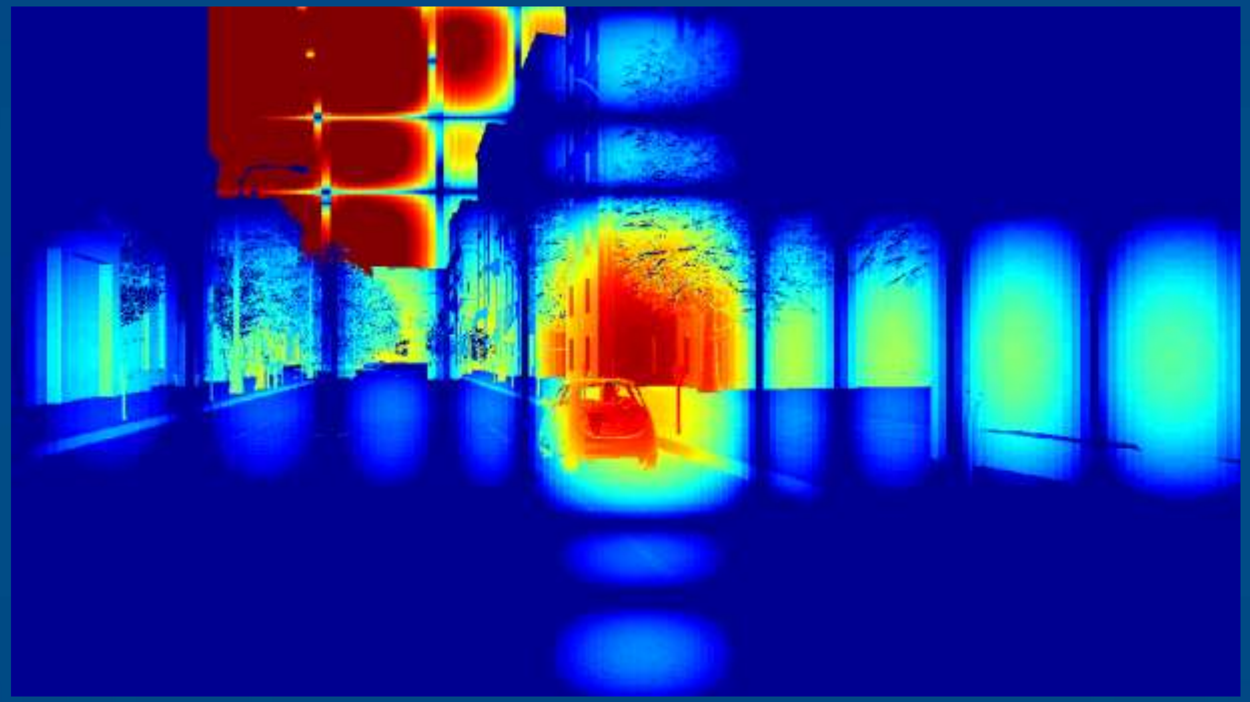


Physical components

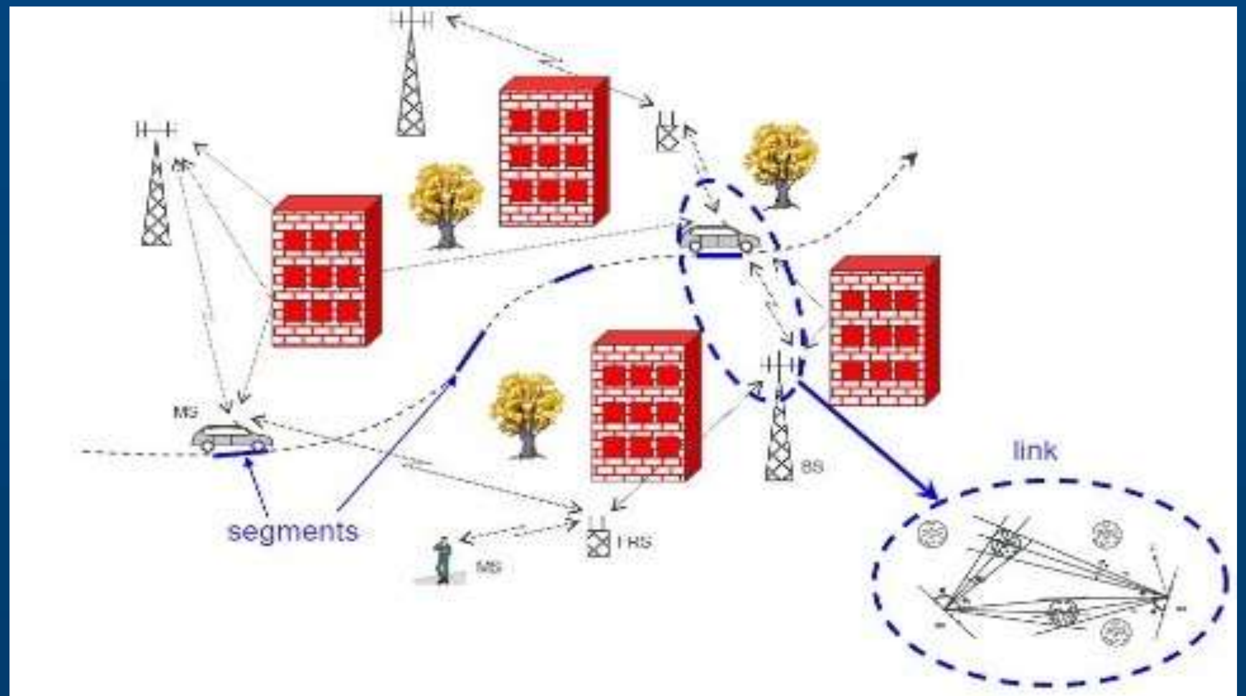
Vehicle Component



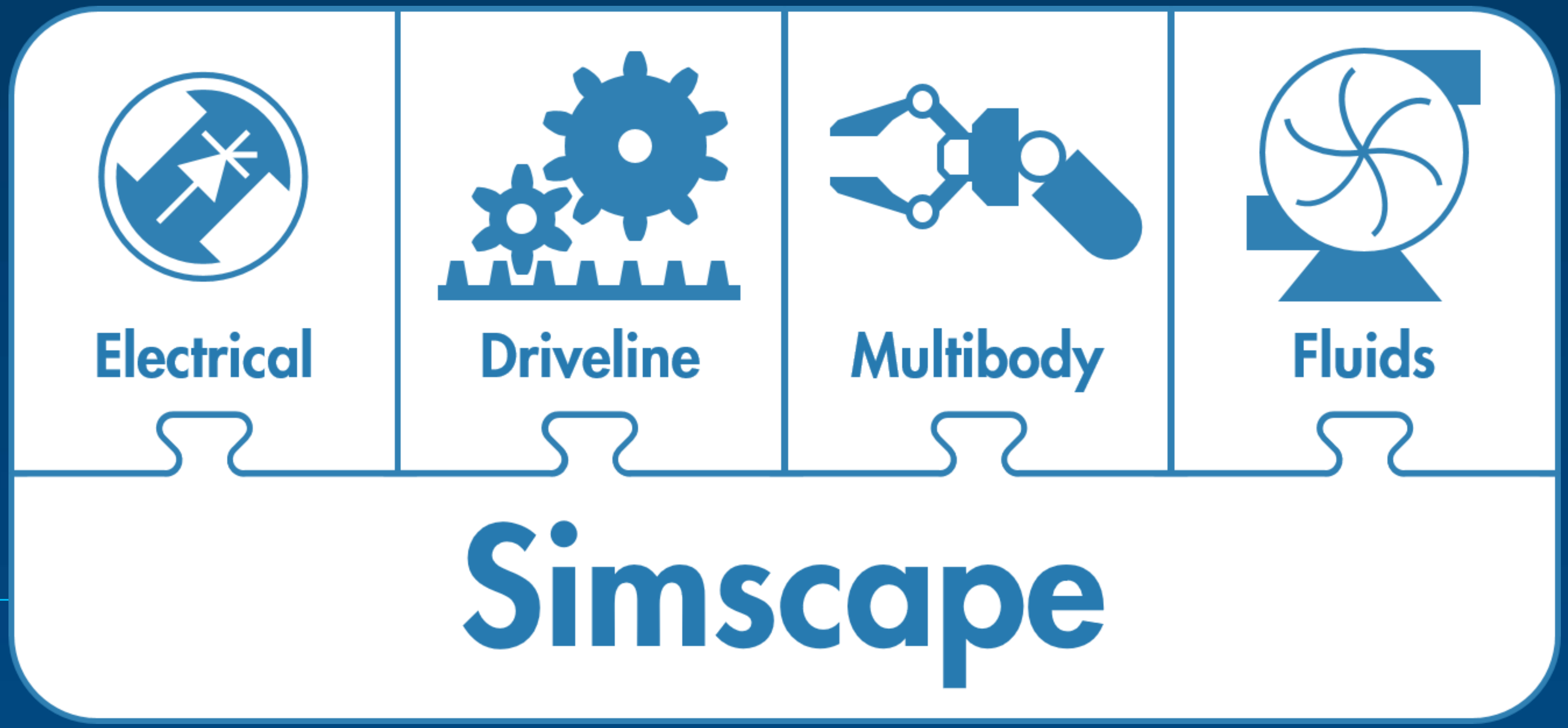
Sensor Model



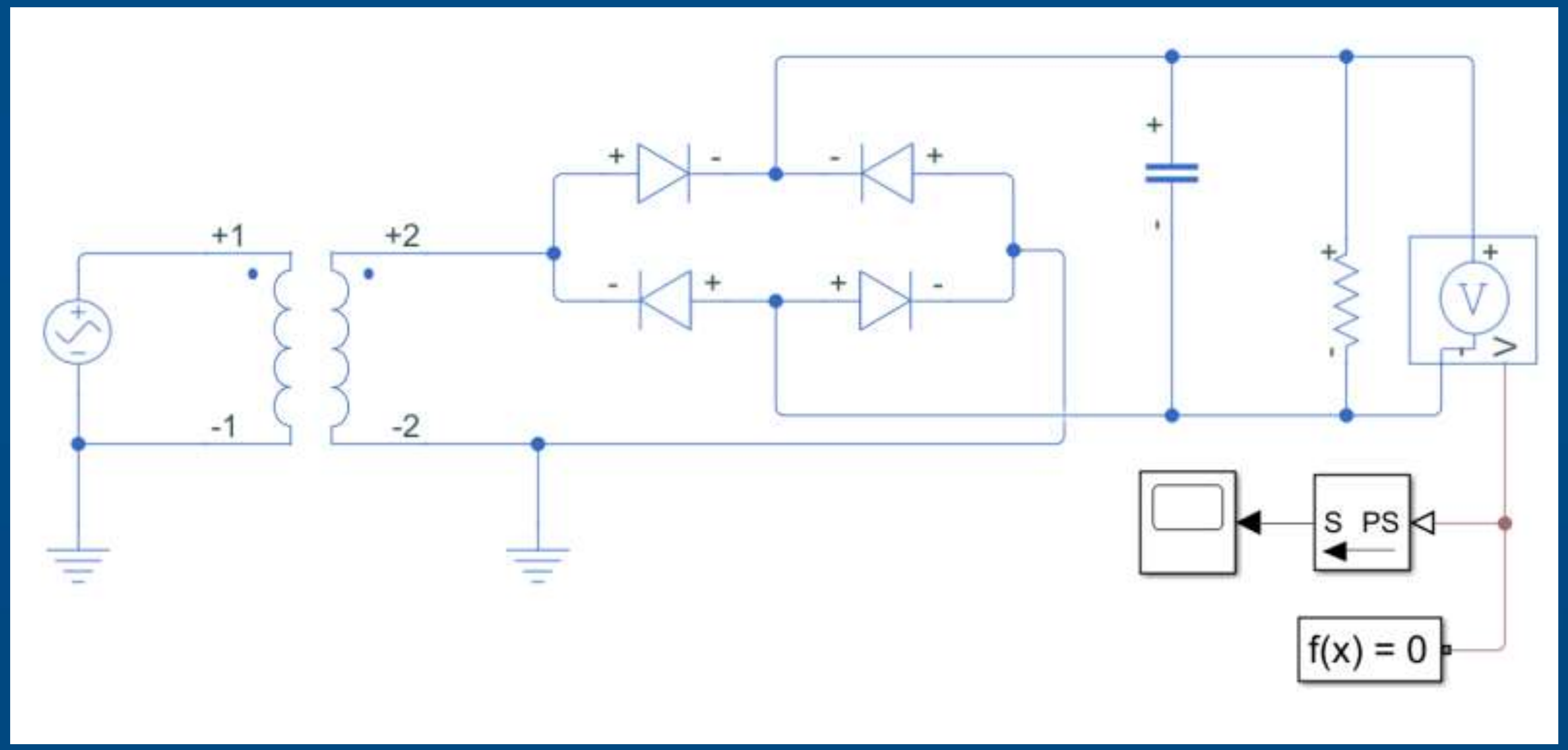
Communications Channel

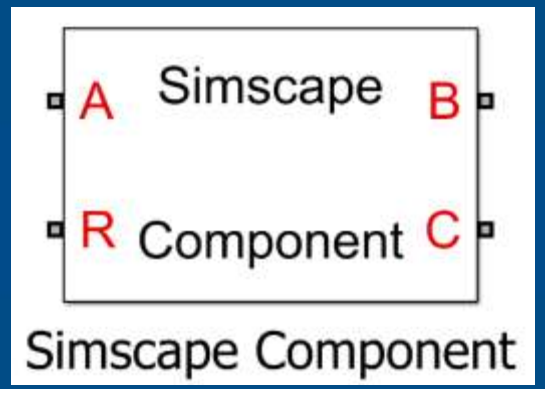


Motor

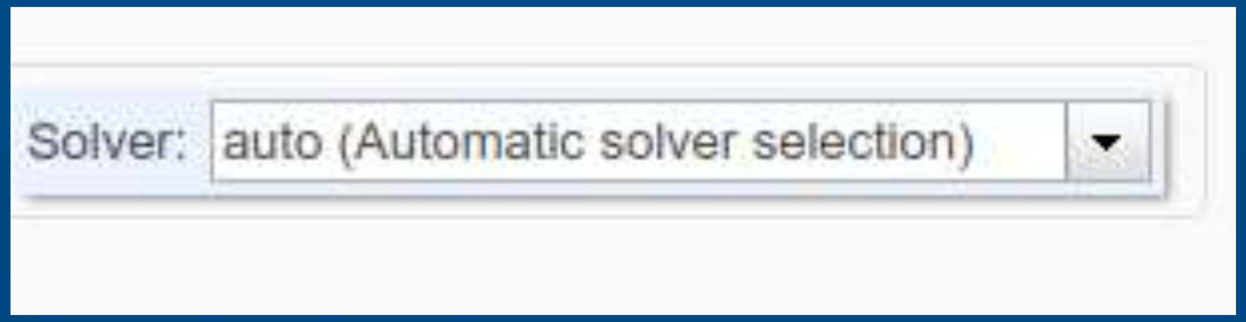


Simscape for physical modeling






```
% 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;
```

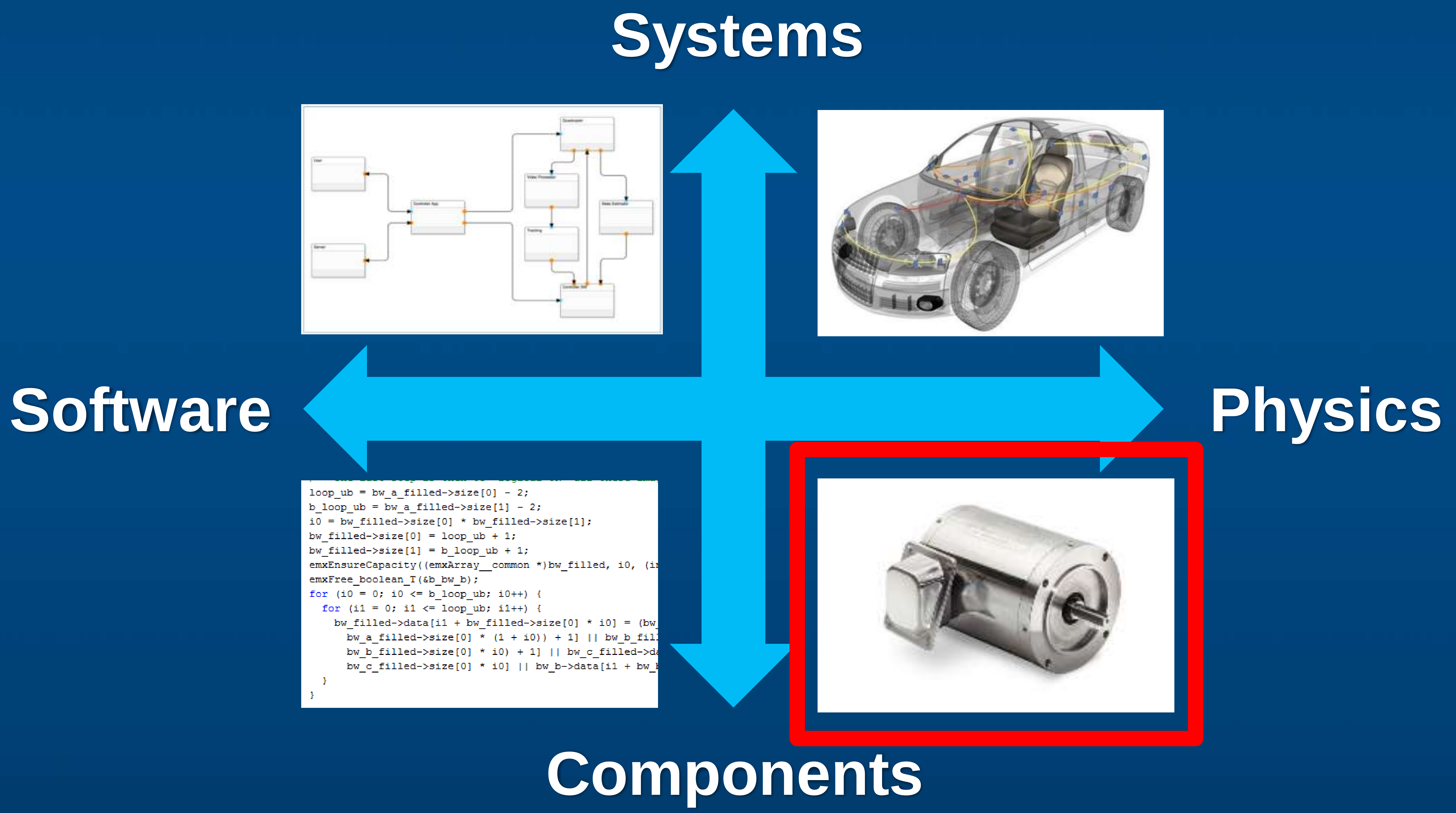


Publication-quality diagrams

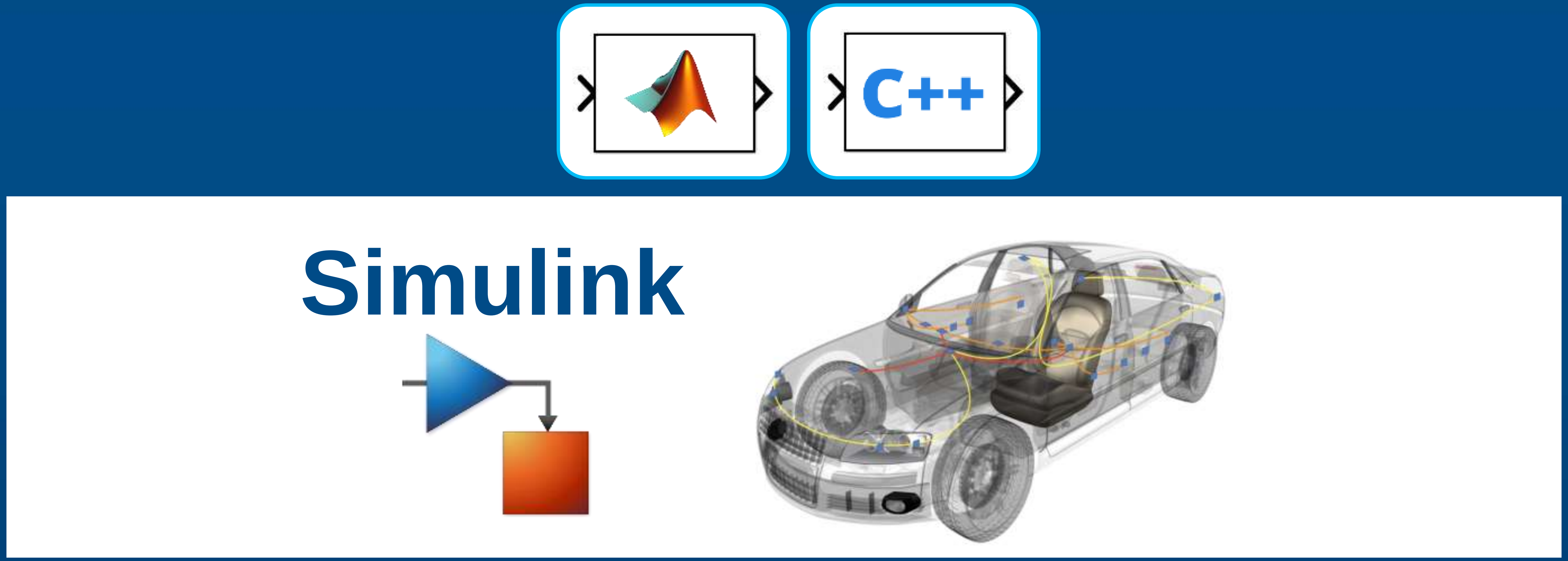
Simscape modeling language

Models just run

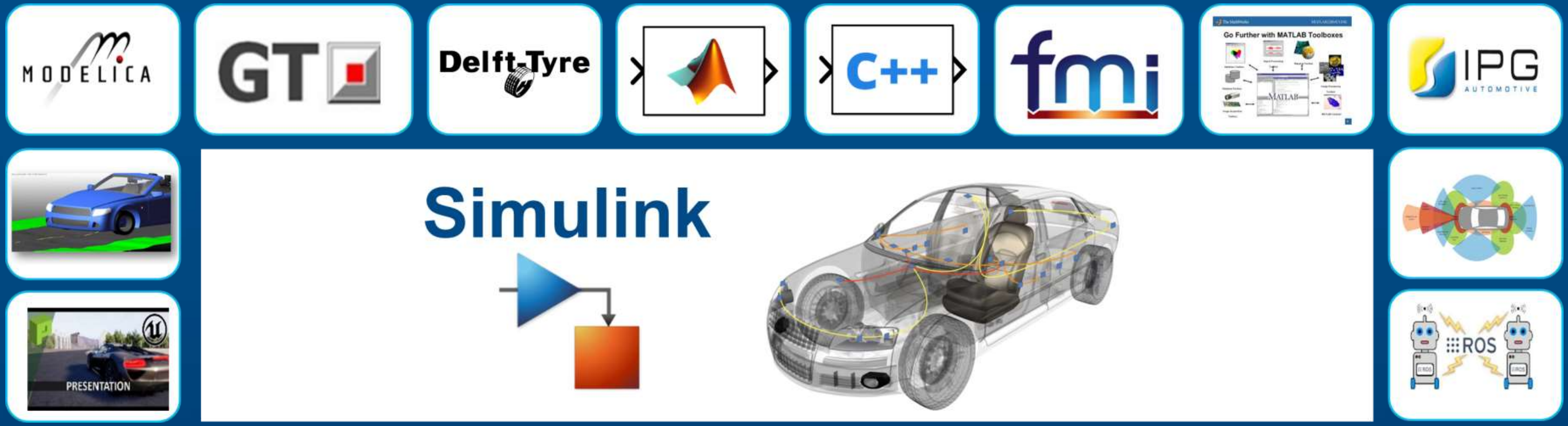
Types of models



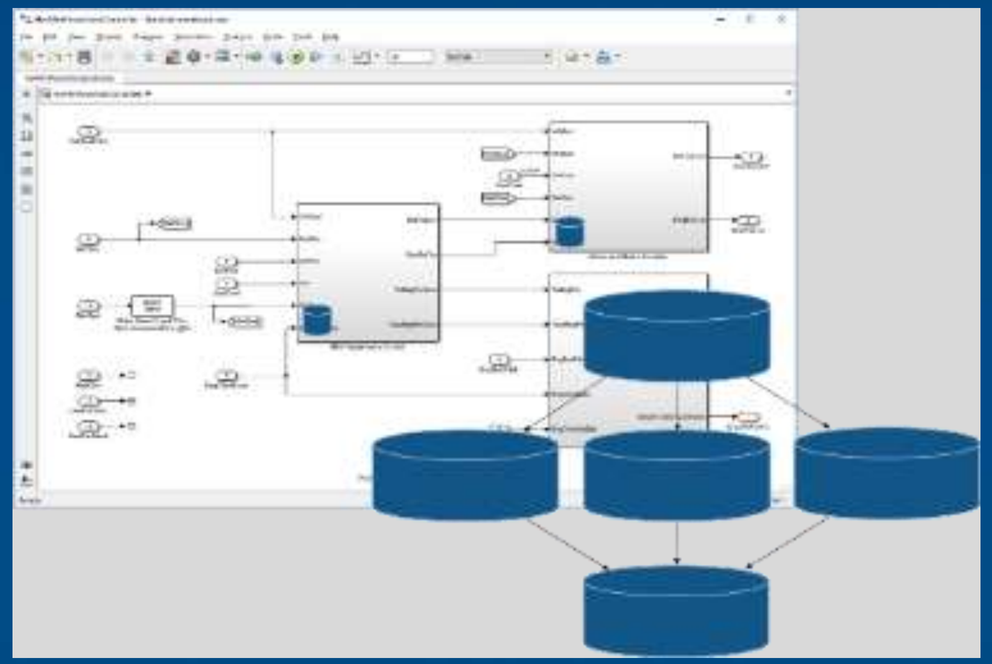
Simulink as an Integration Platform



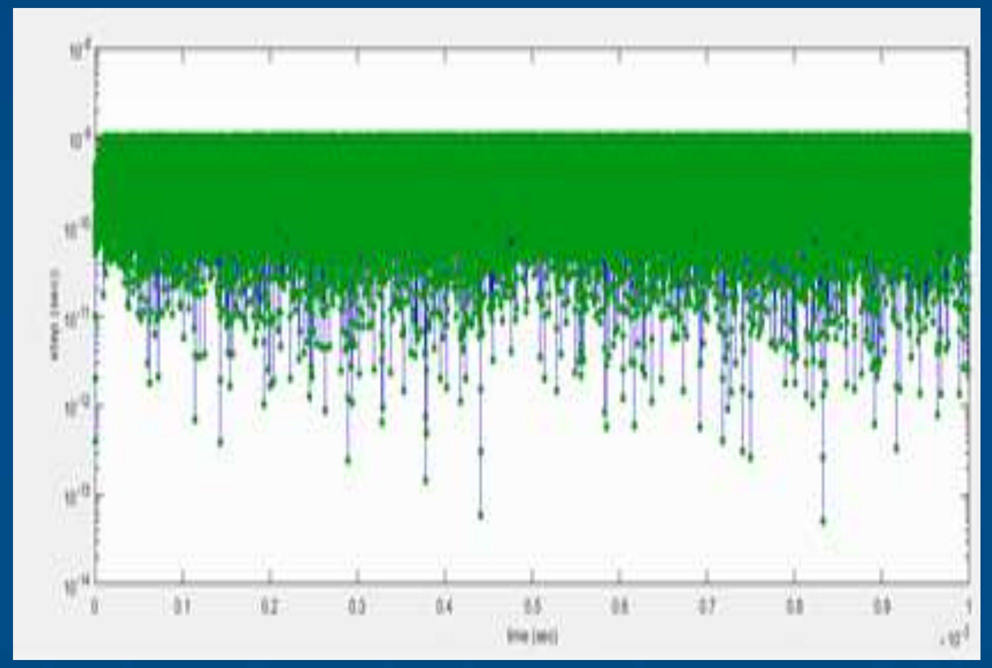
Simulink as an Integration Platform



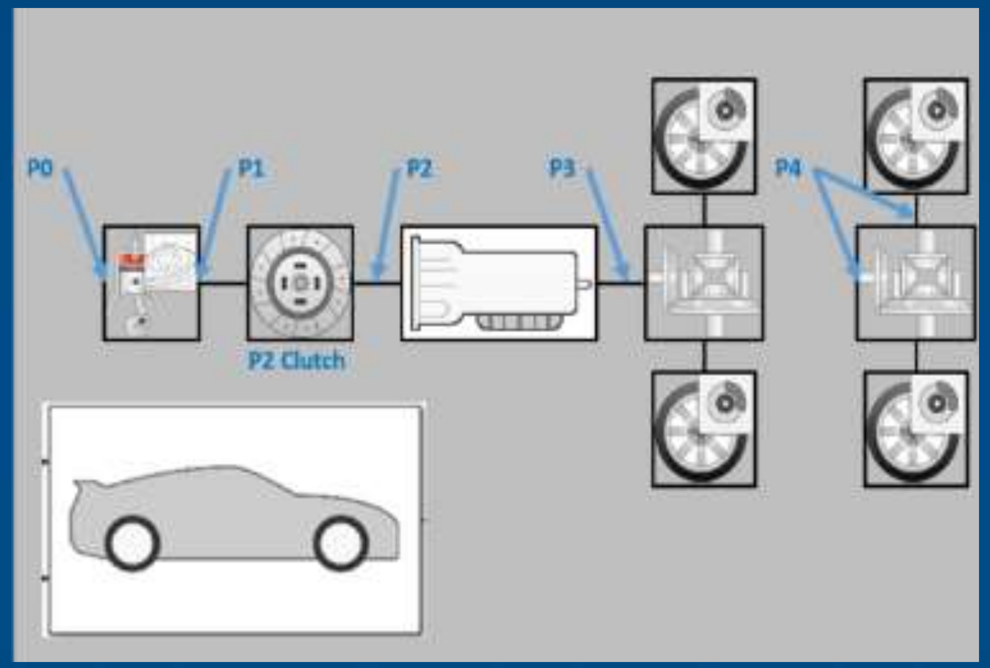
Simulation Integration: Infrastructure



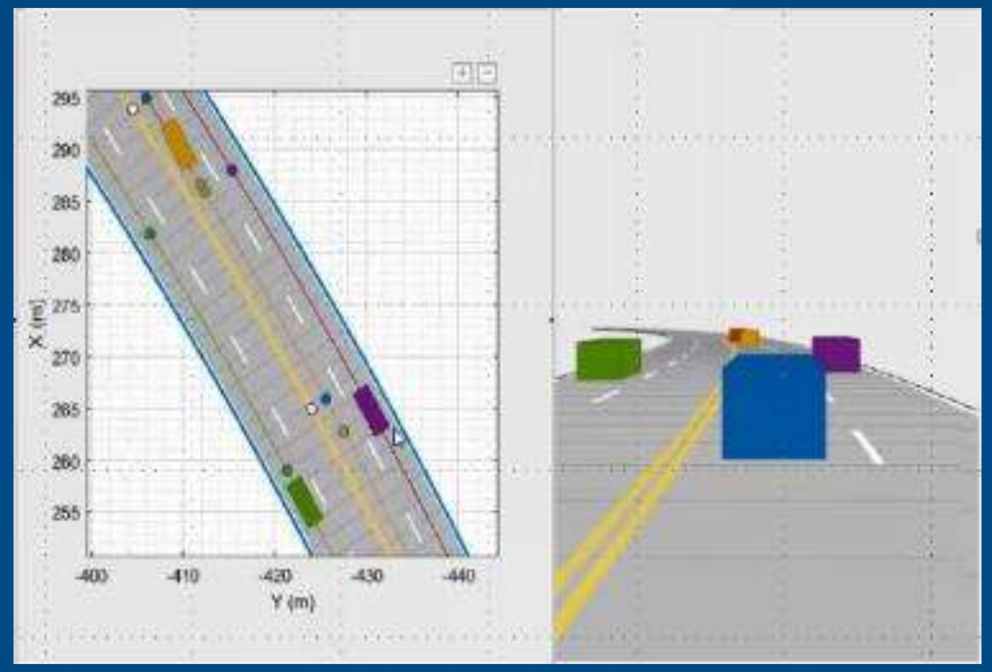
Data Management



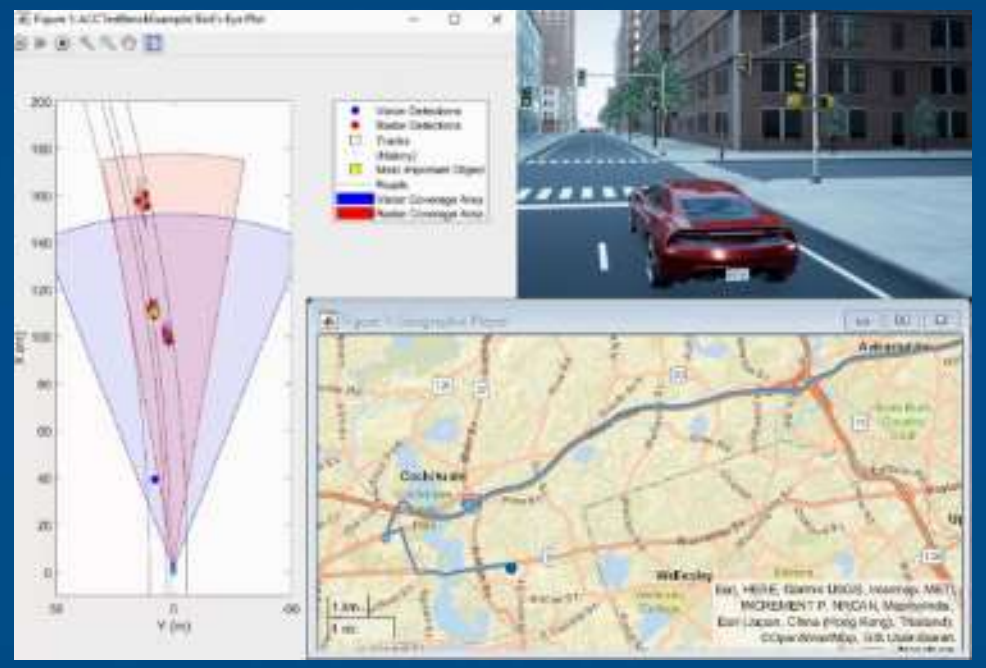
Solver Technology



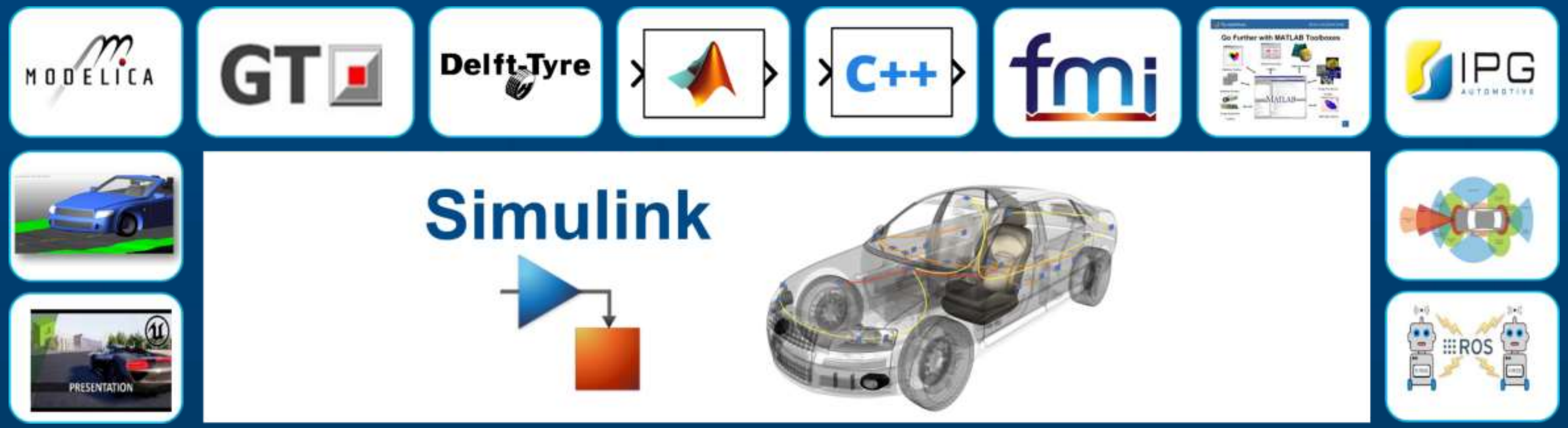
Vehicle Configuration



Multi-actor Scenarios



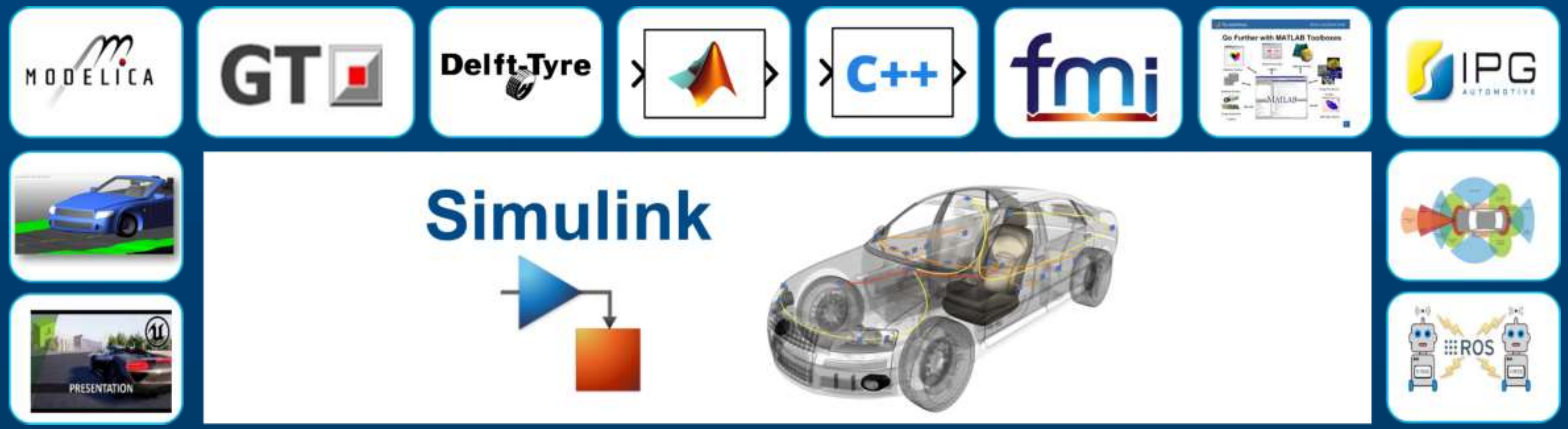
Visualization



Simulation Integration: Analyses

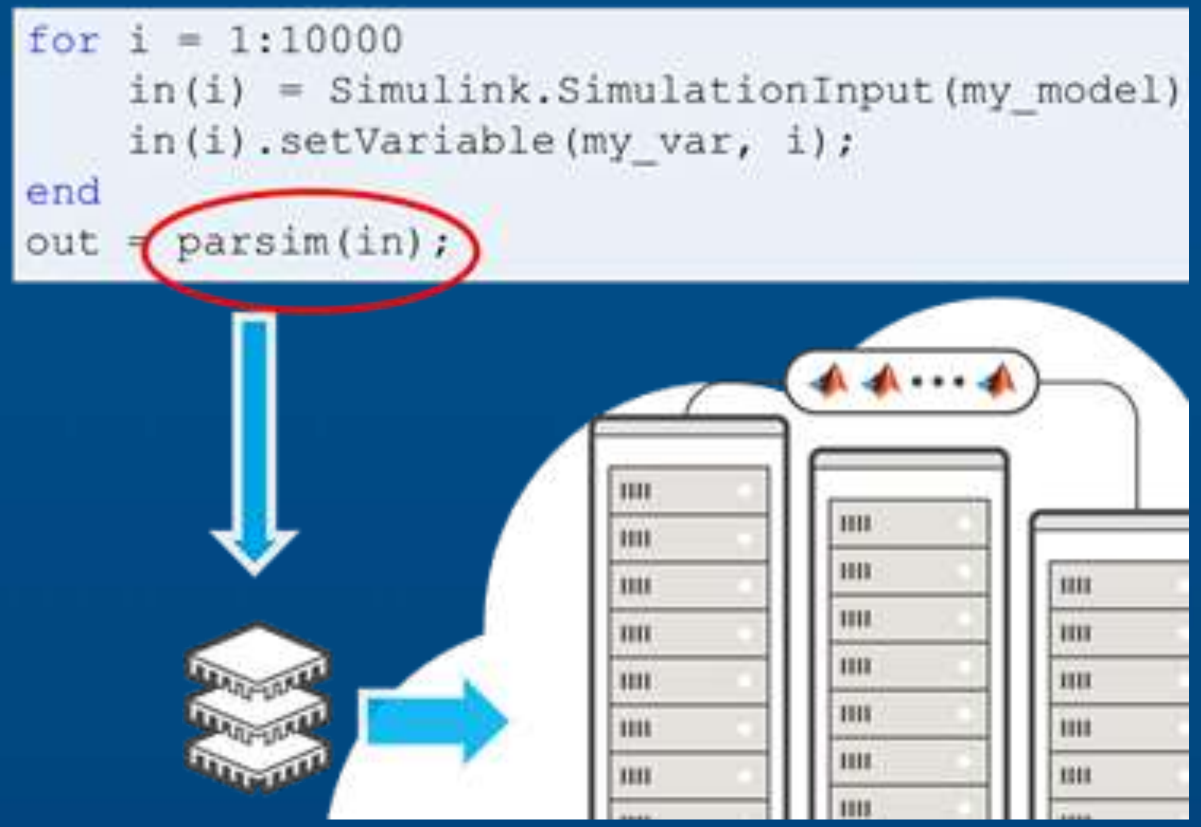
Verification and Validation | Design Optimization | Sensitivity Analysis | Virtual Calibration

Fuel Economy | Performance | Energy Consumption | Drivability | Ride & Handling

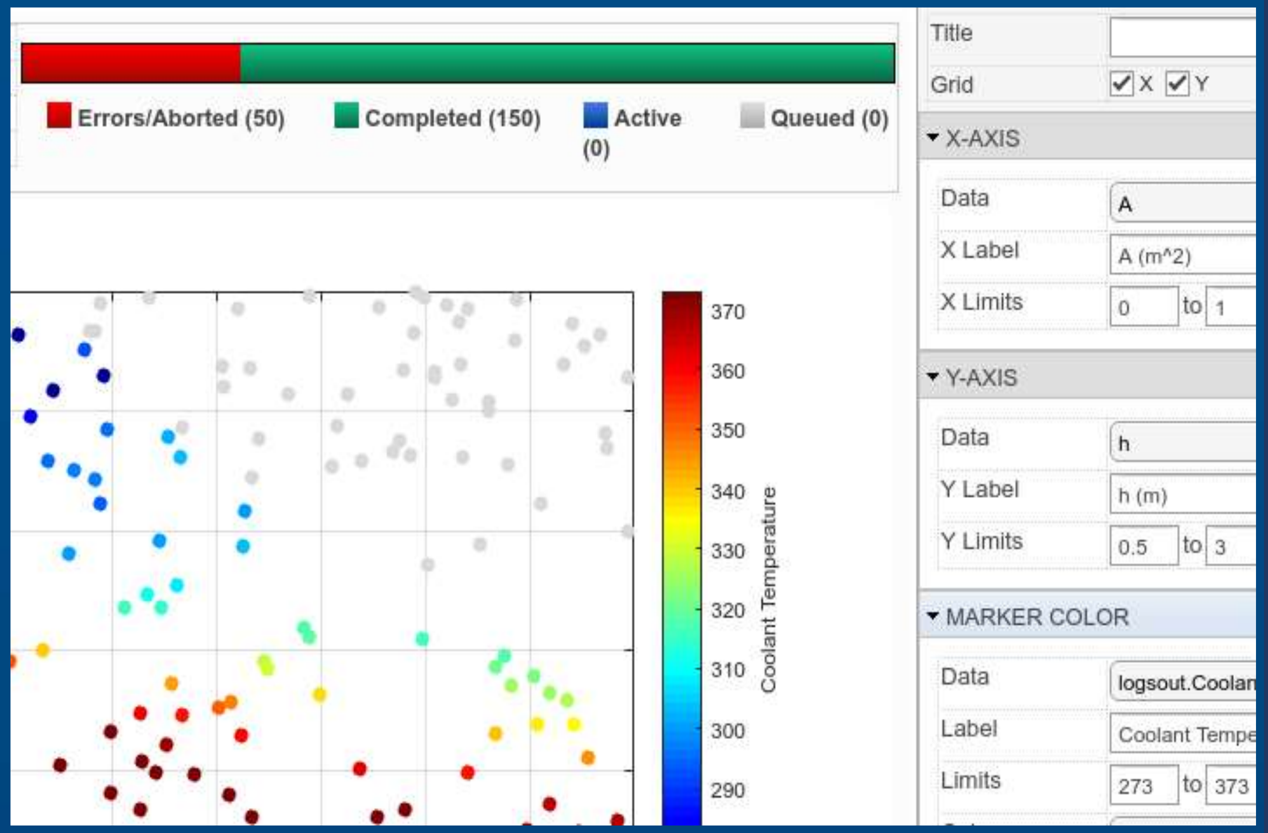


Scaling up simulations

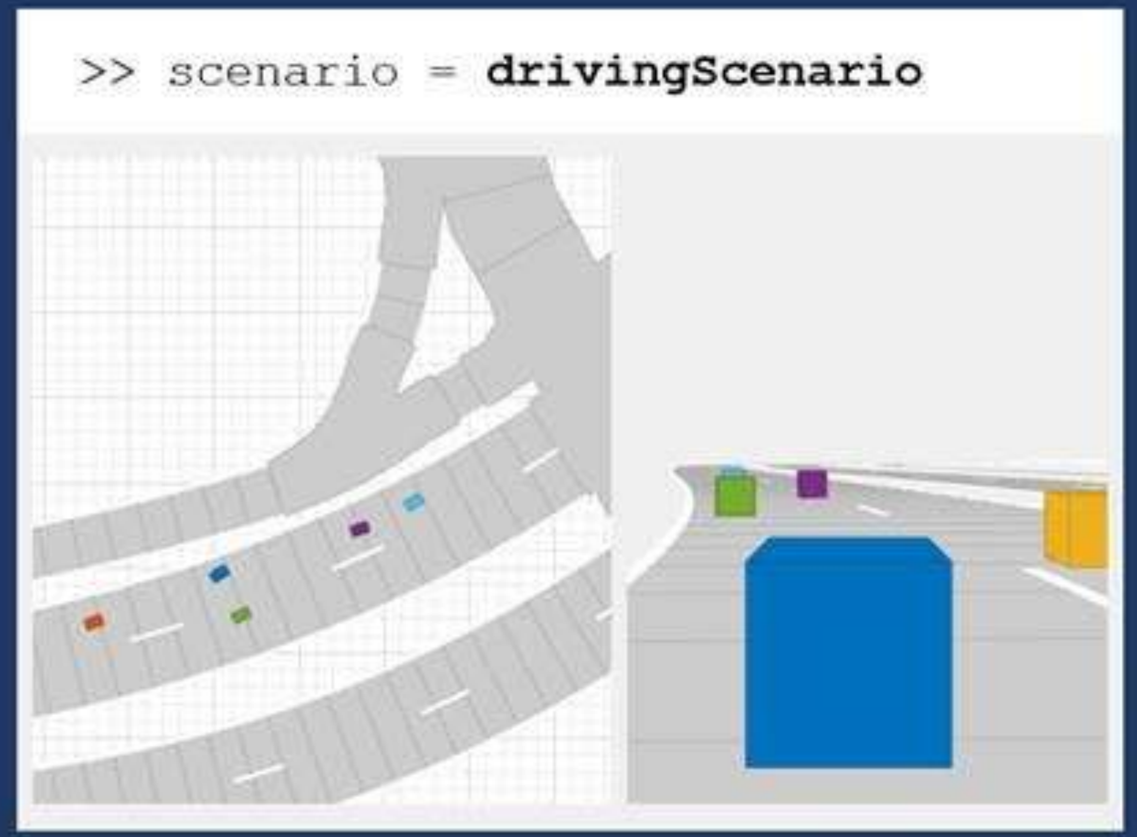
X 1,000,000’s



Parallel simulations



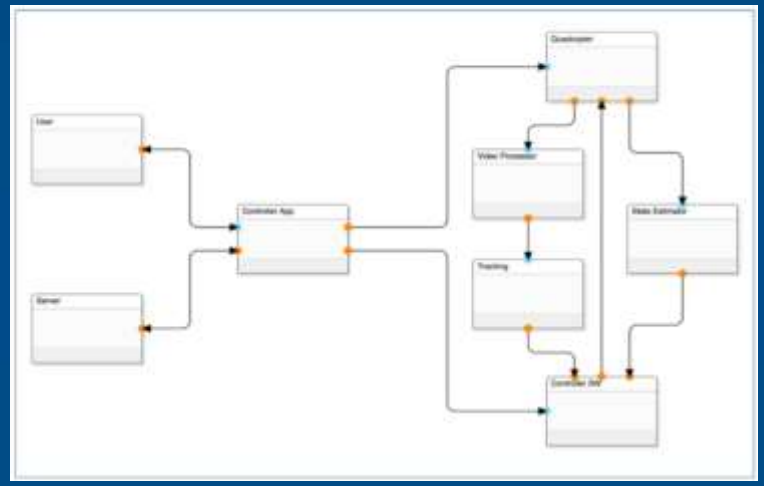
Simulation Manager

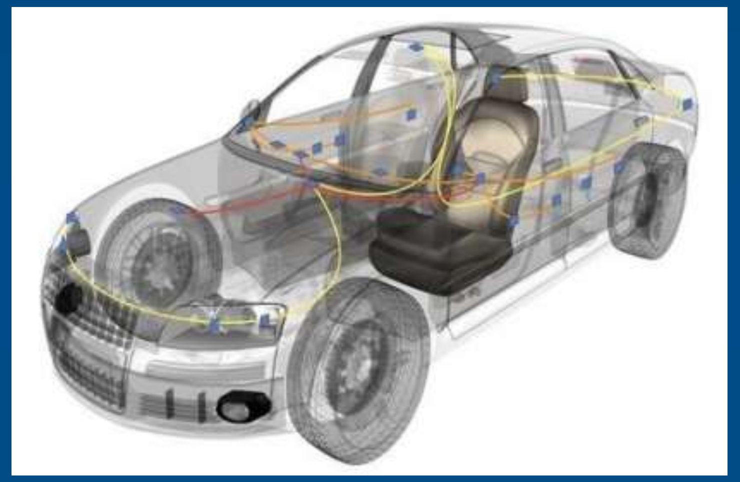


Programmatic test creation

Types of models

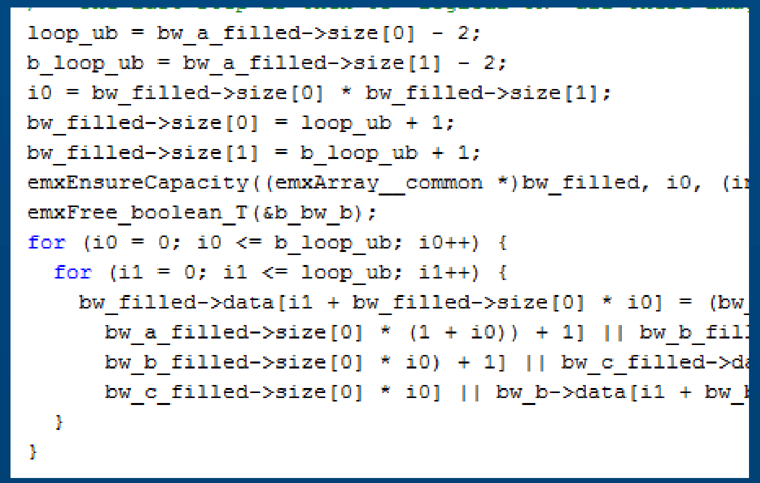
Systems





Software

Physics

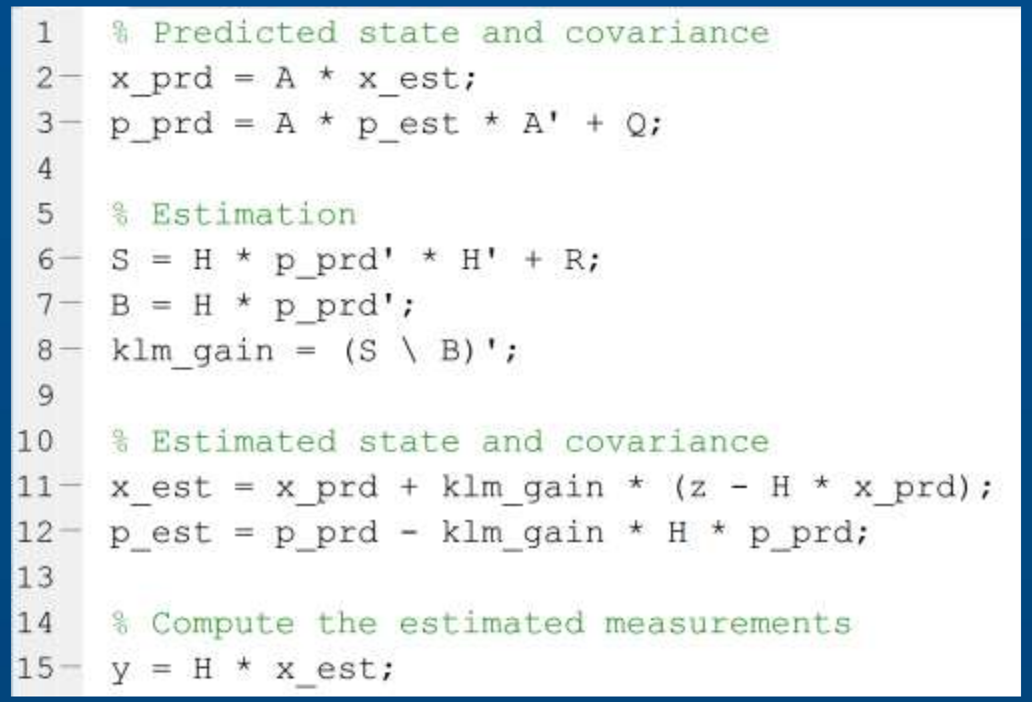


```
loop_ub = bw_a_filled->size[0] - 2;
b_loop_ub = bw_a_filled->size[1] - 2;
i0 = bw_filled->size[0] * bw_filled->size[1];
bw_filled->size[0] = loop_ub + 1;
bw_filled->size[1] = b_loop_ub + 1;
emxEnsureCapacity((emxArray__common *)bw_filled, i0, (i
emxFree_boolean_T(&b_bw_b);
for (i0 = 0; i0 <= b_loop_ub;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

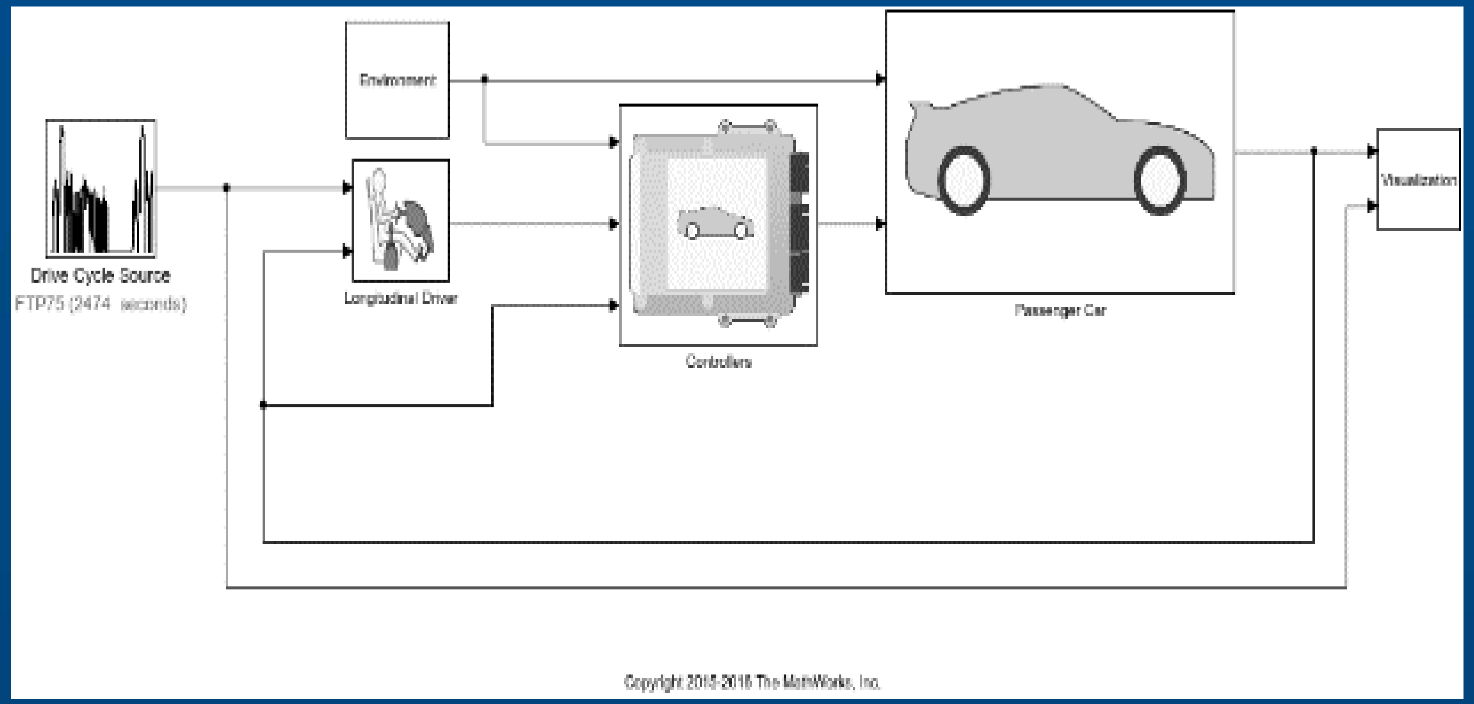
Components

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

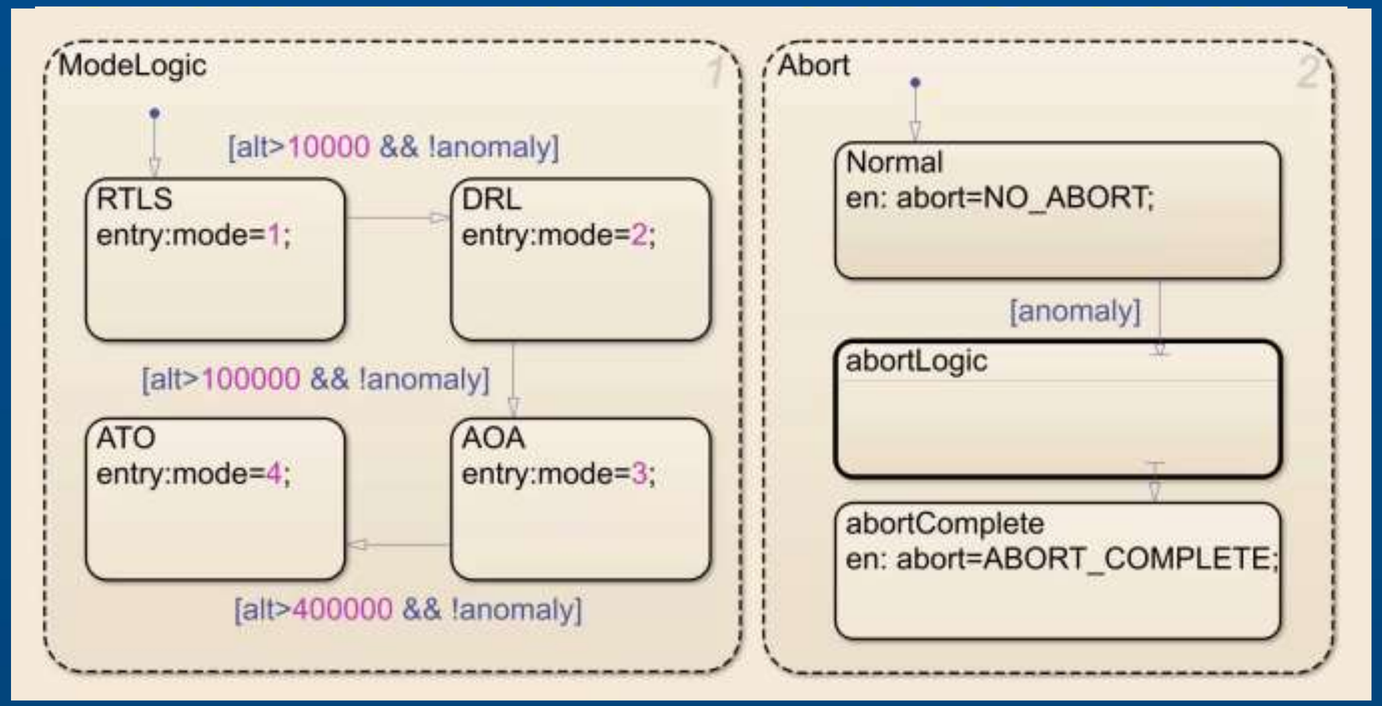
Working at a high-level of abstraction



MATLAB

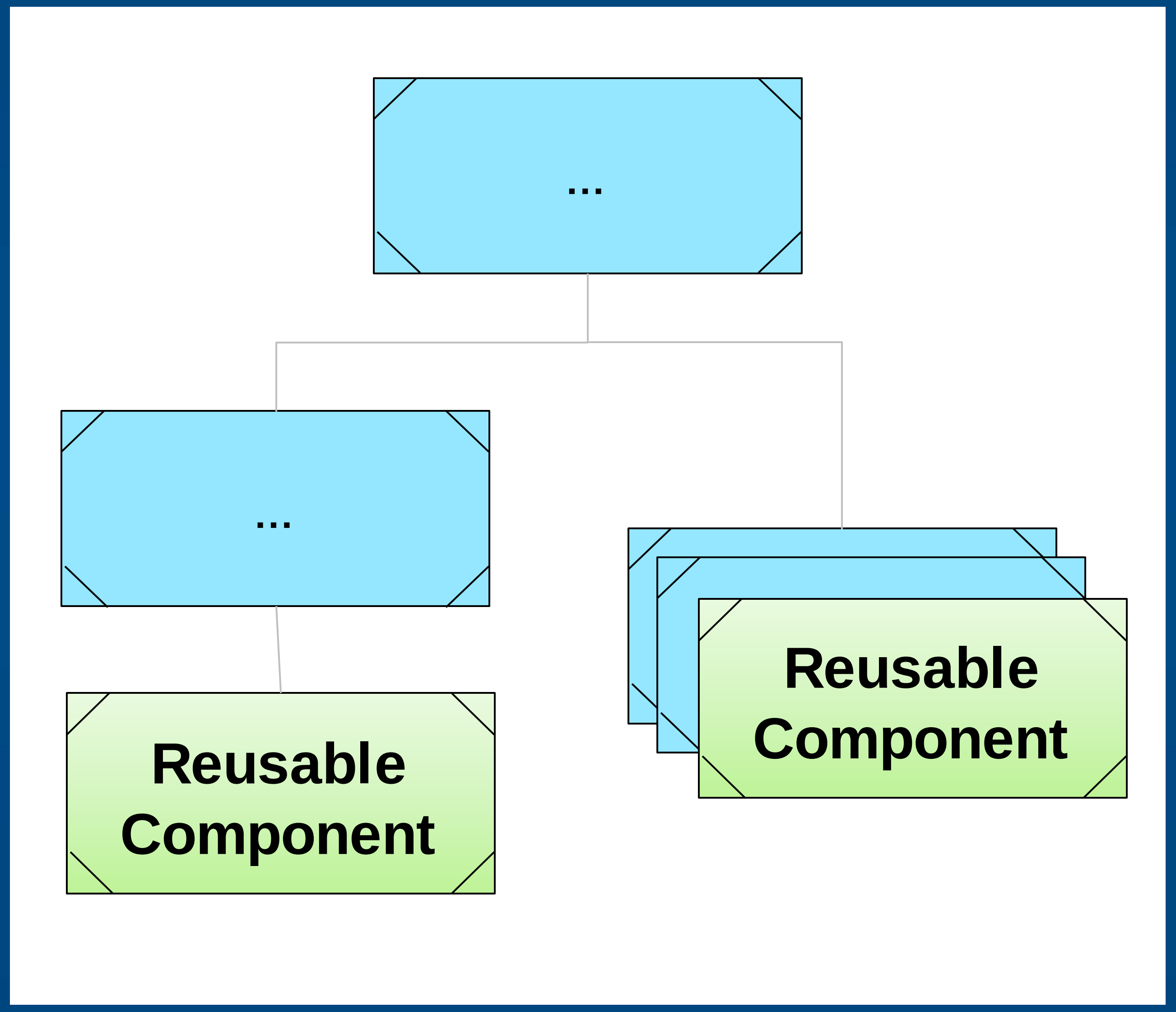


Simulink

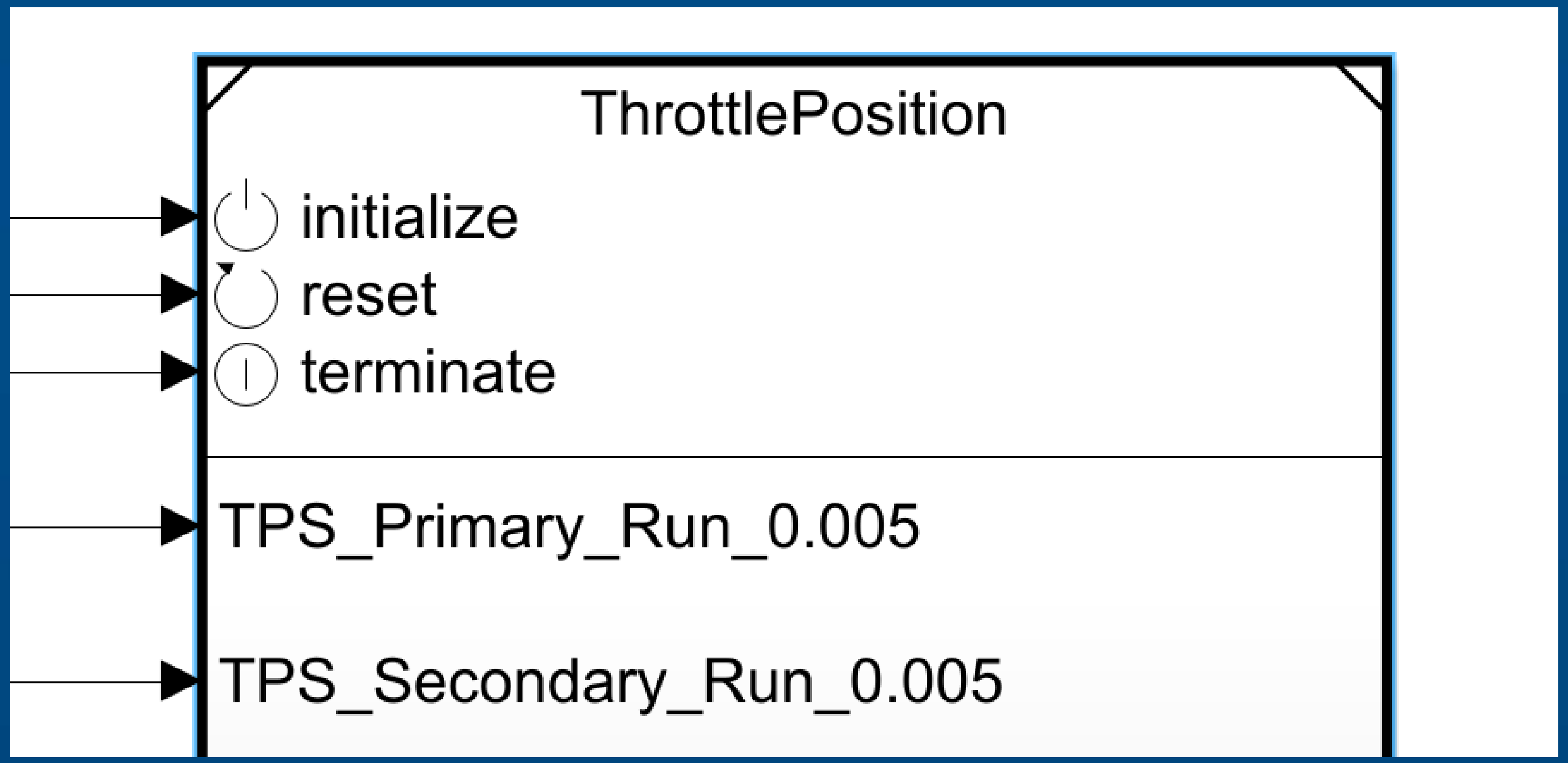


Stateflow

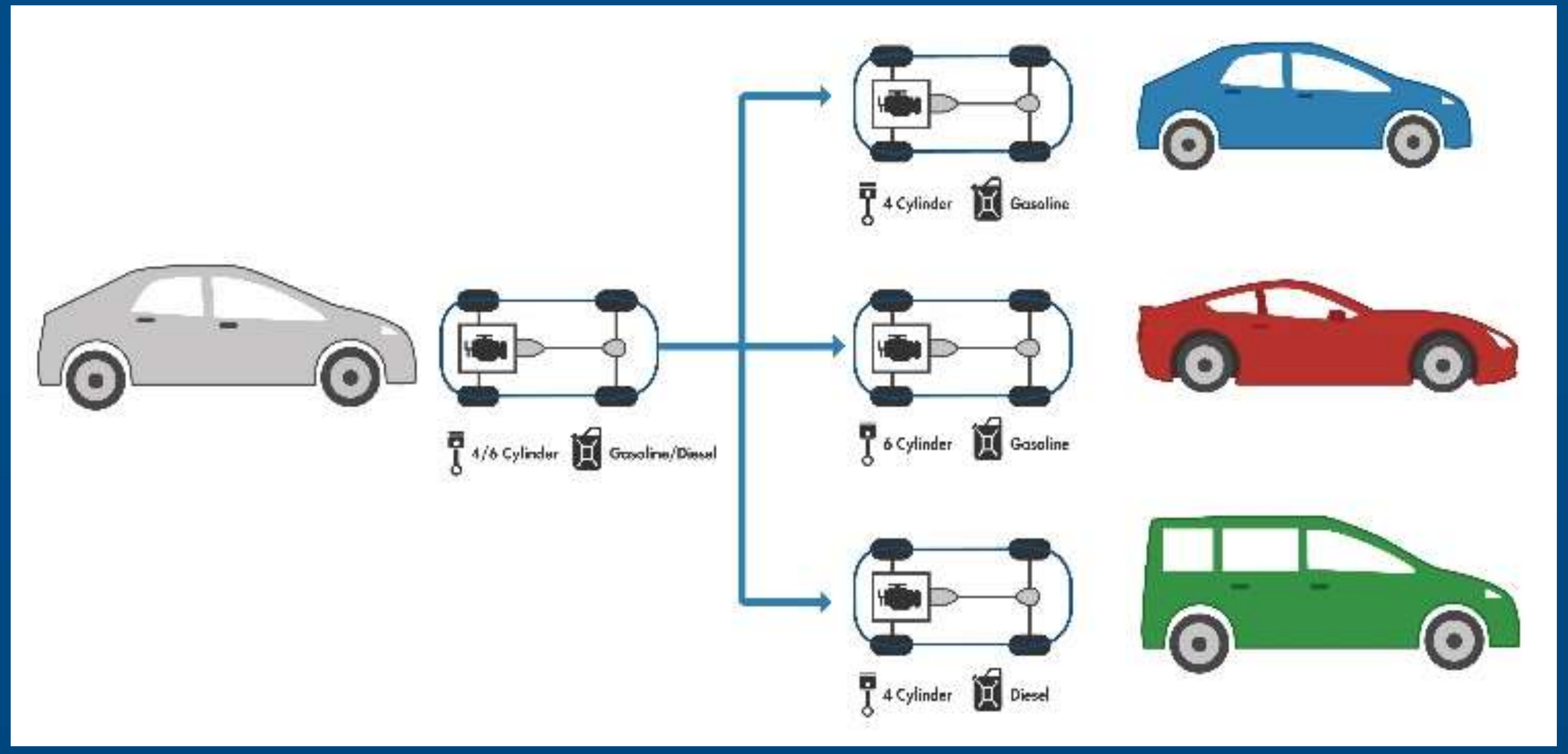
Component modeling



Reusable components that can be adapted to any software system

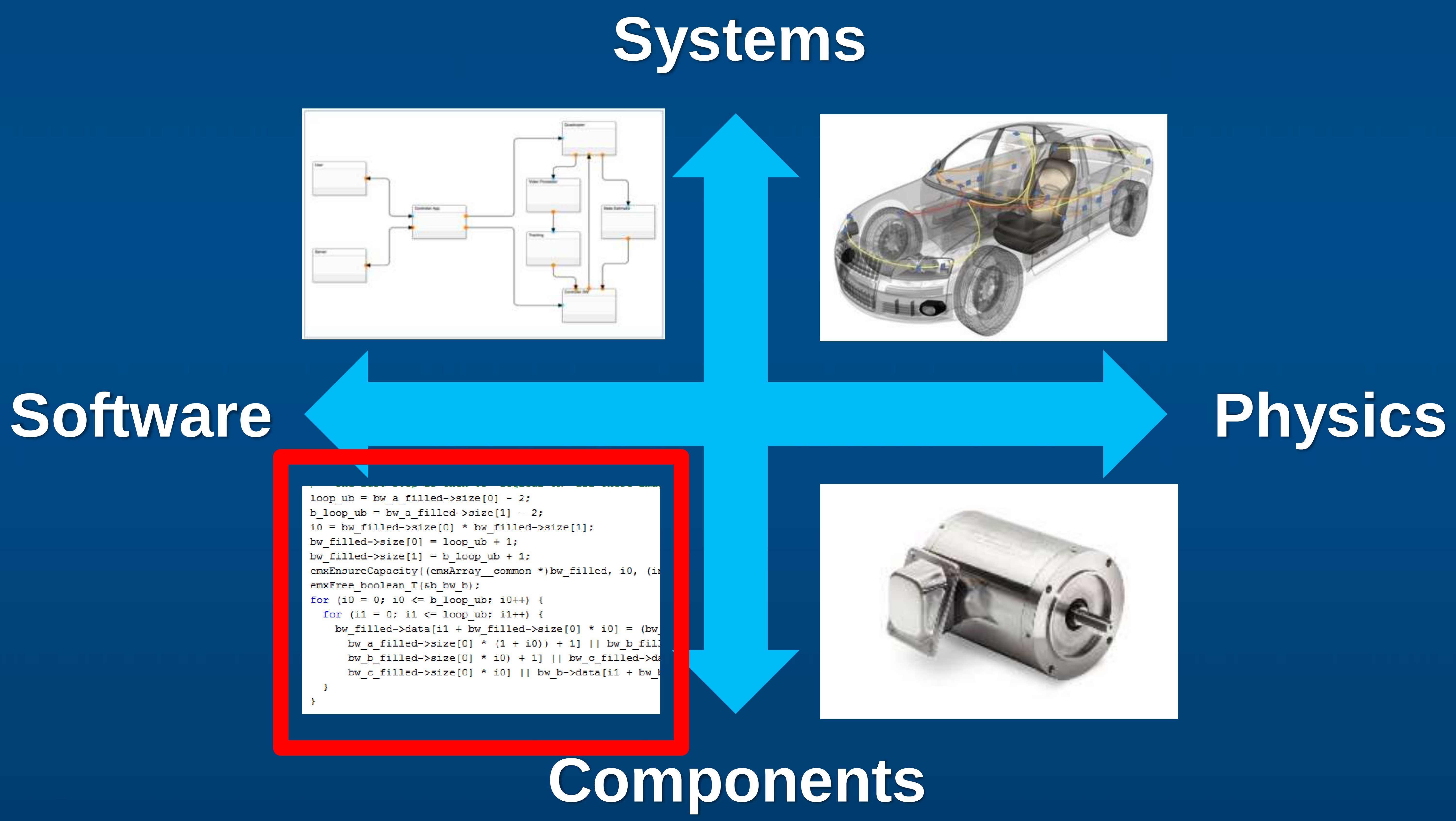


Startup and shutdown behavior



Variant management

Types of models



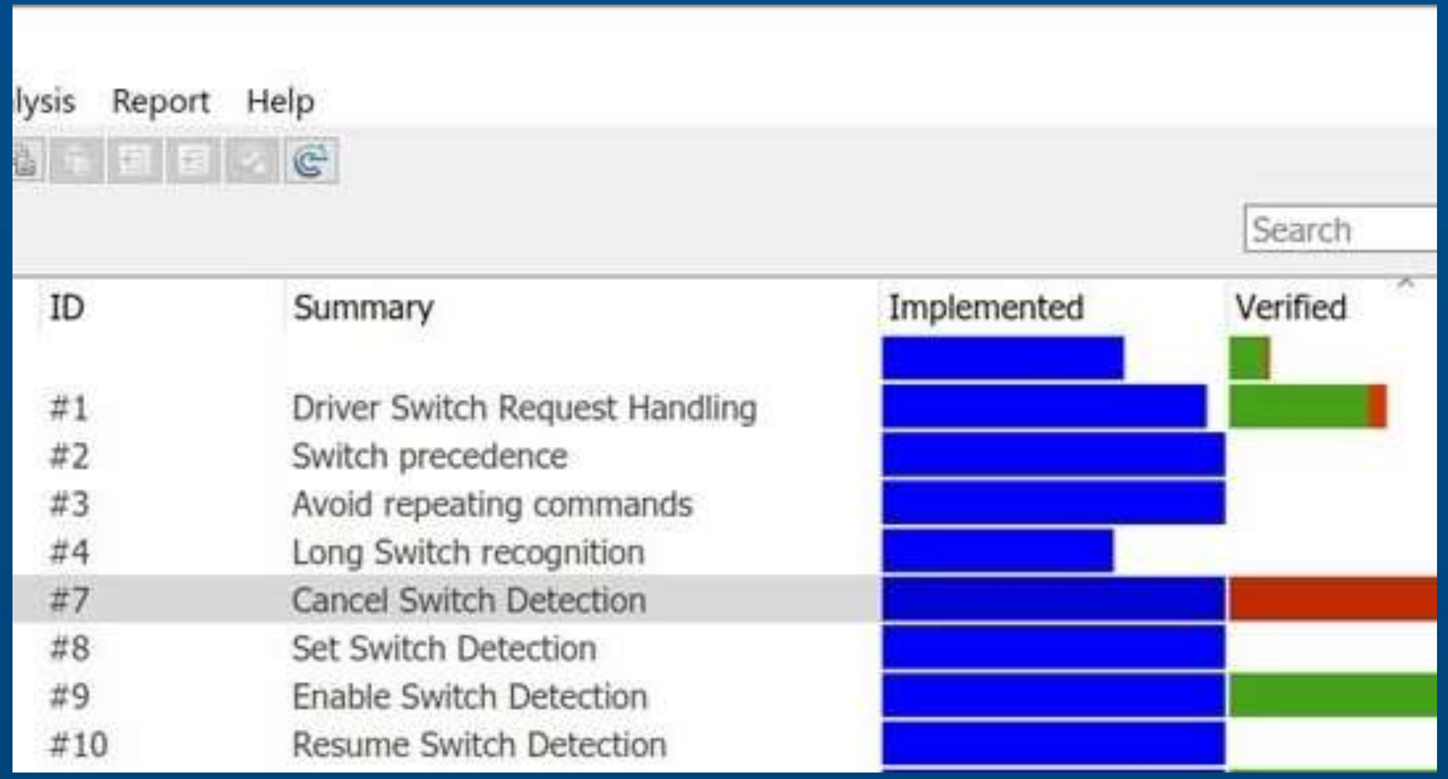
System architecture is the #1 topic

Breakout Topic Requests (2018)

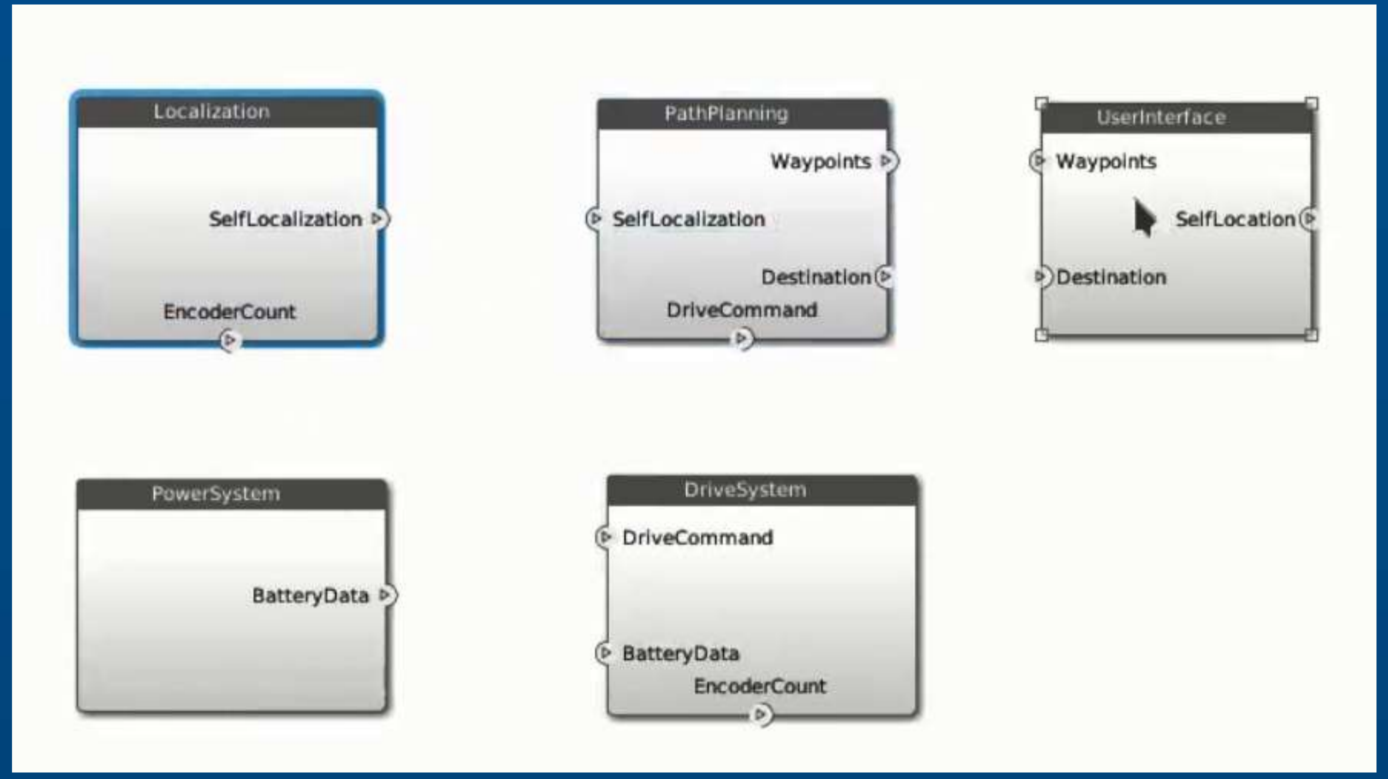
Feature Prioritization (2017)

Systems engineering

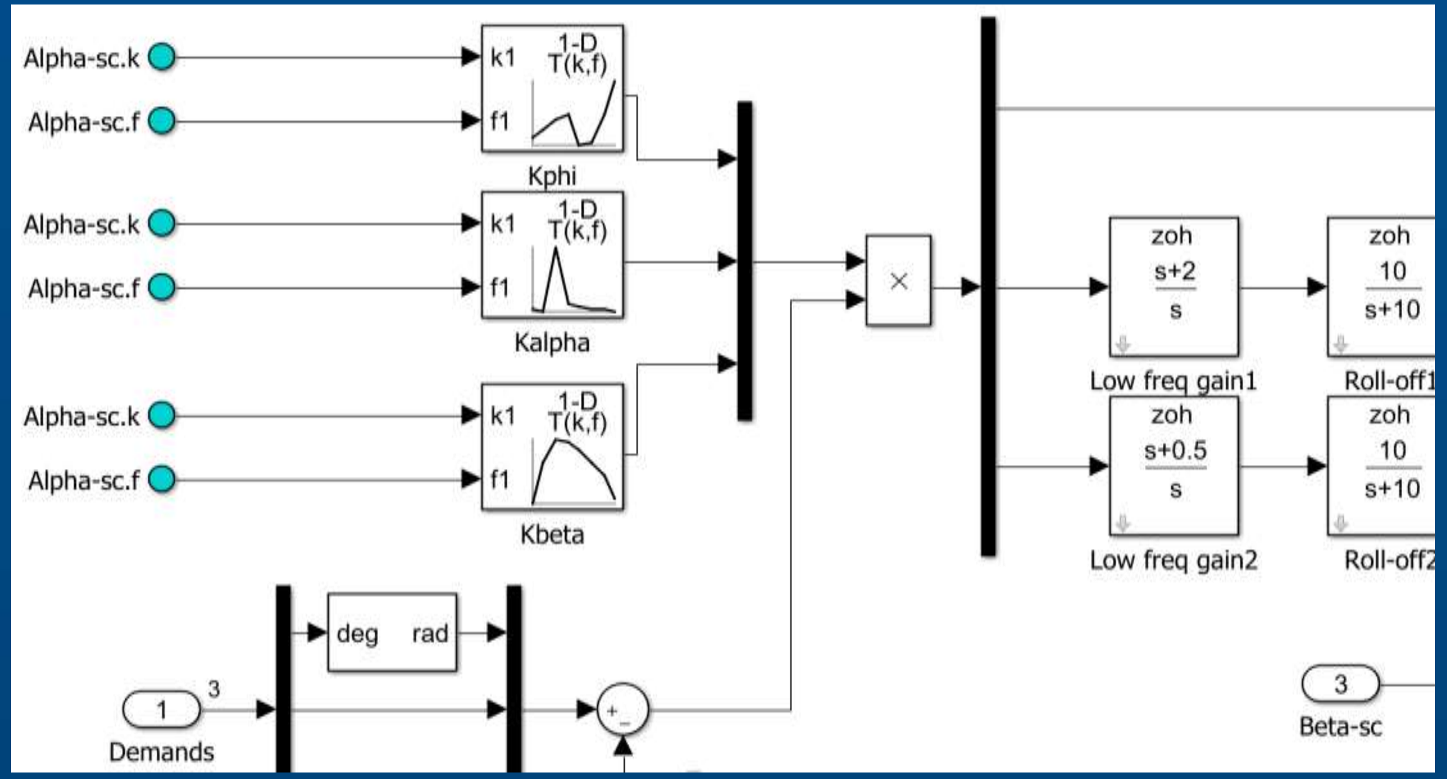
Requirements



Systems



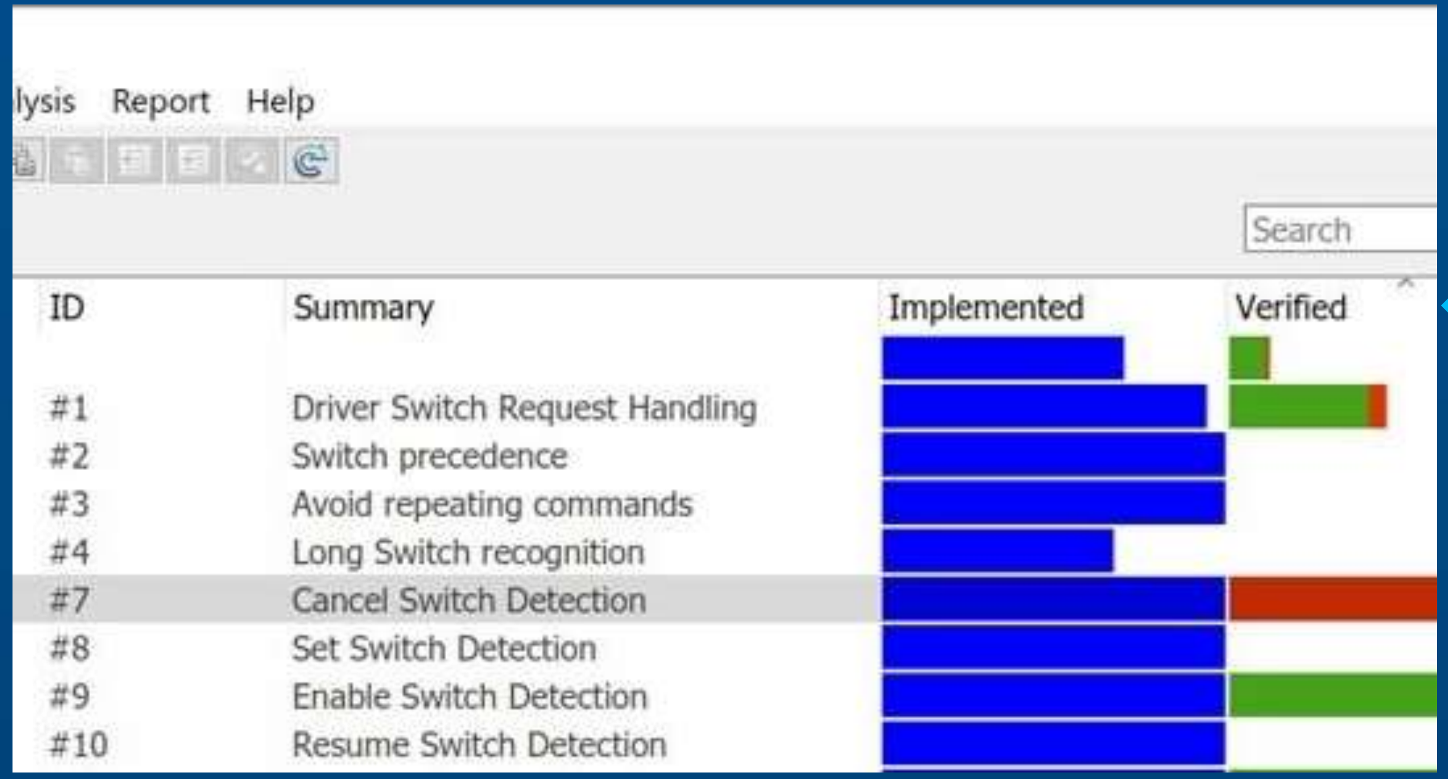
Components

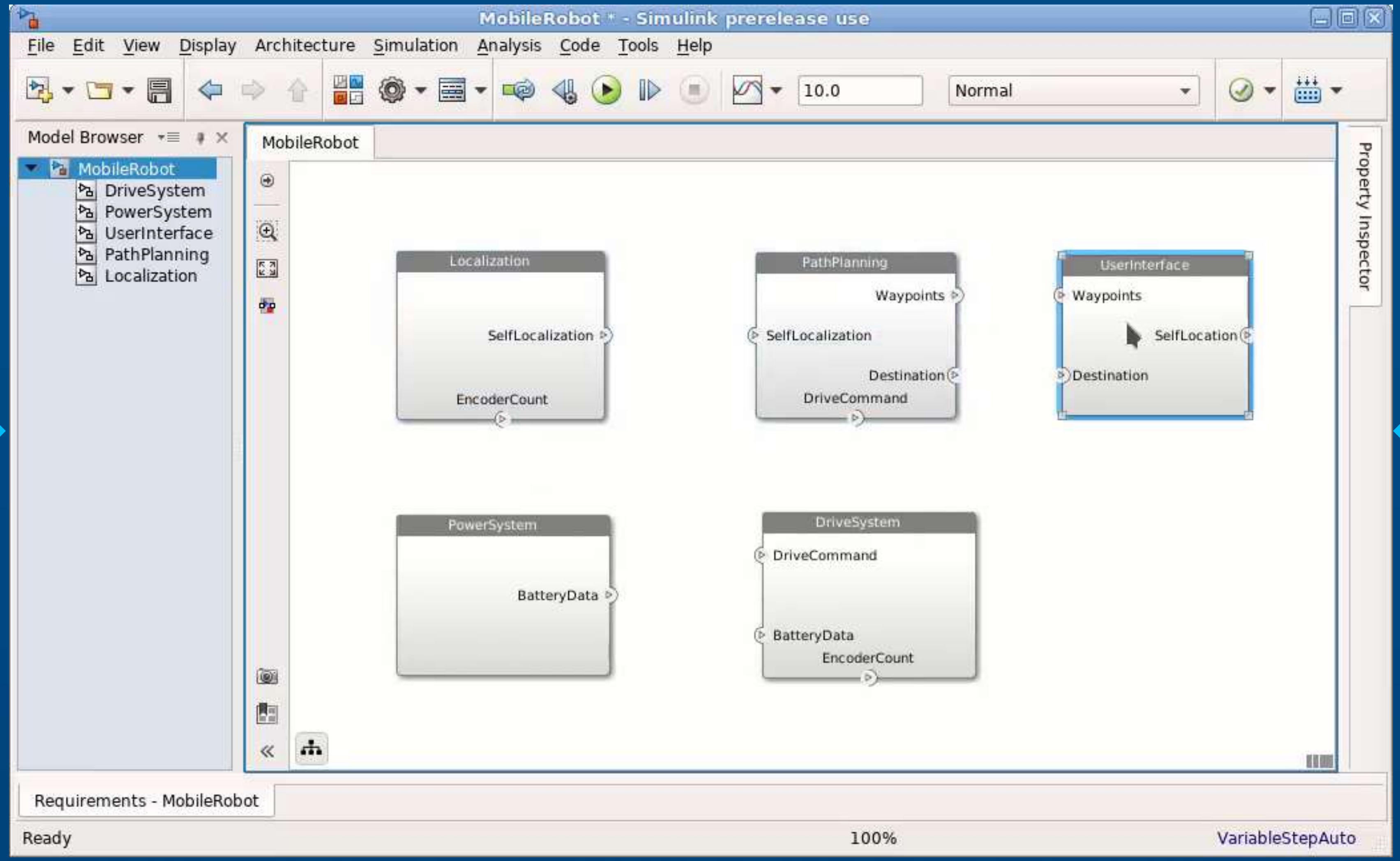


Systems engineering

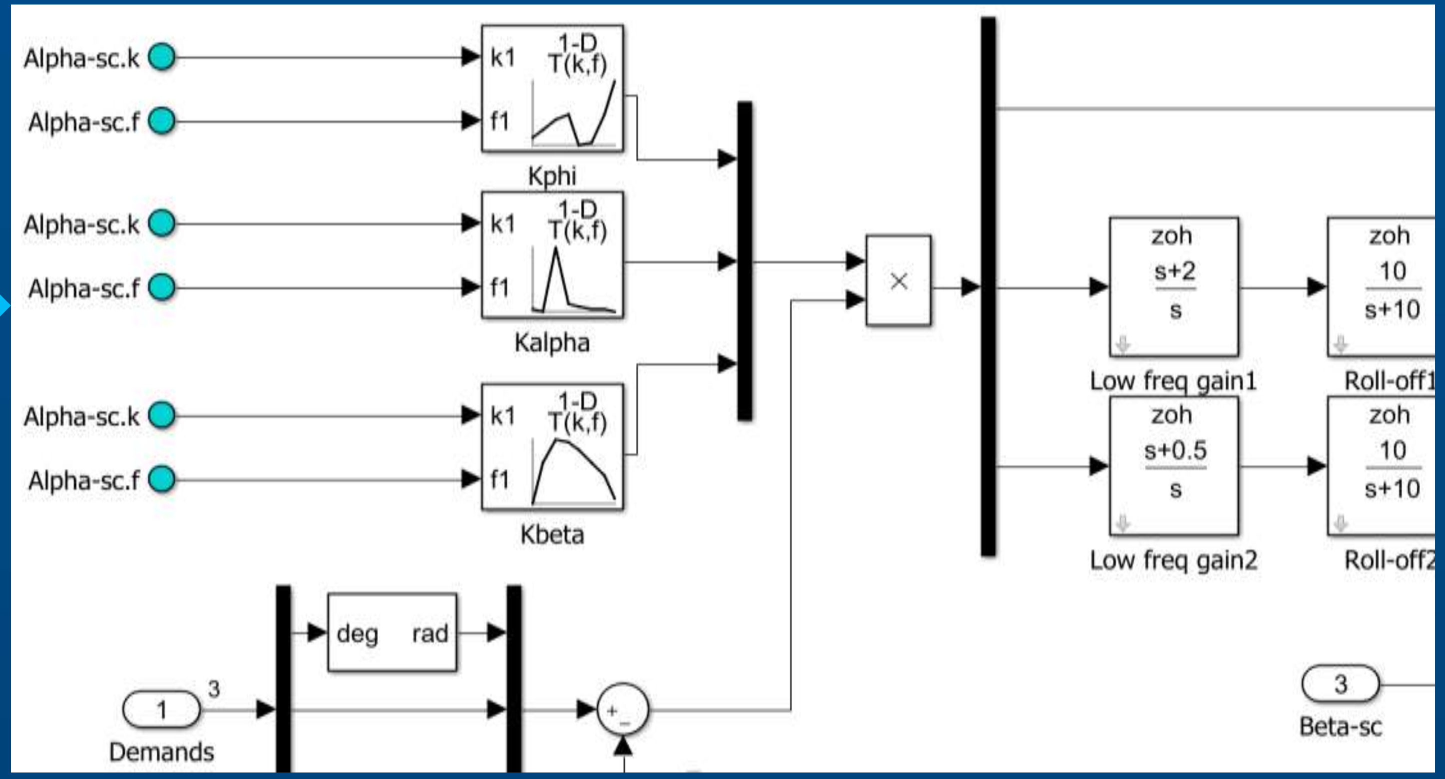
R2019a
System Composer

Requirements

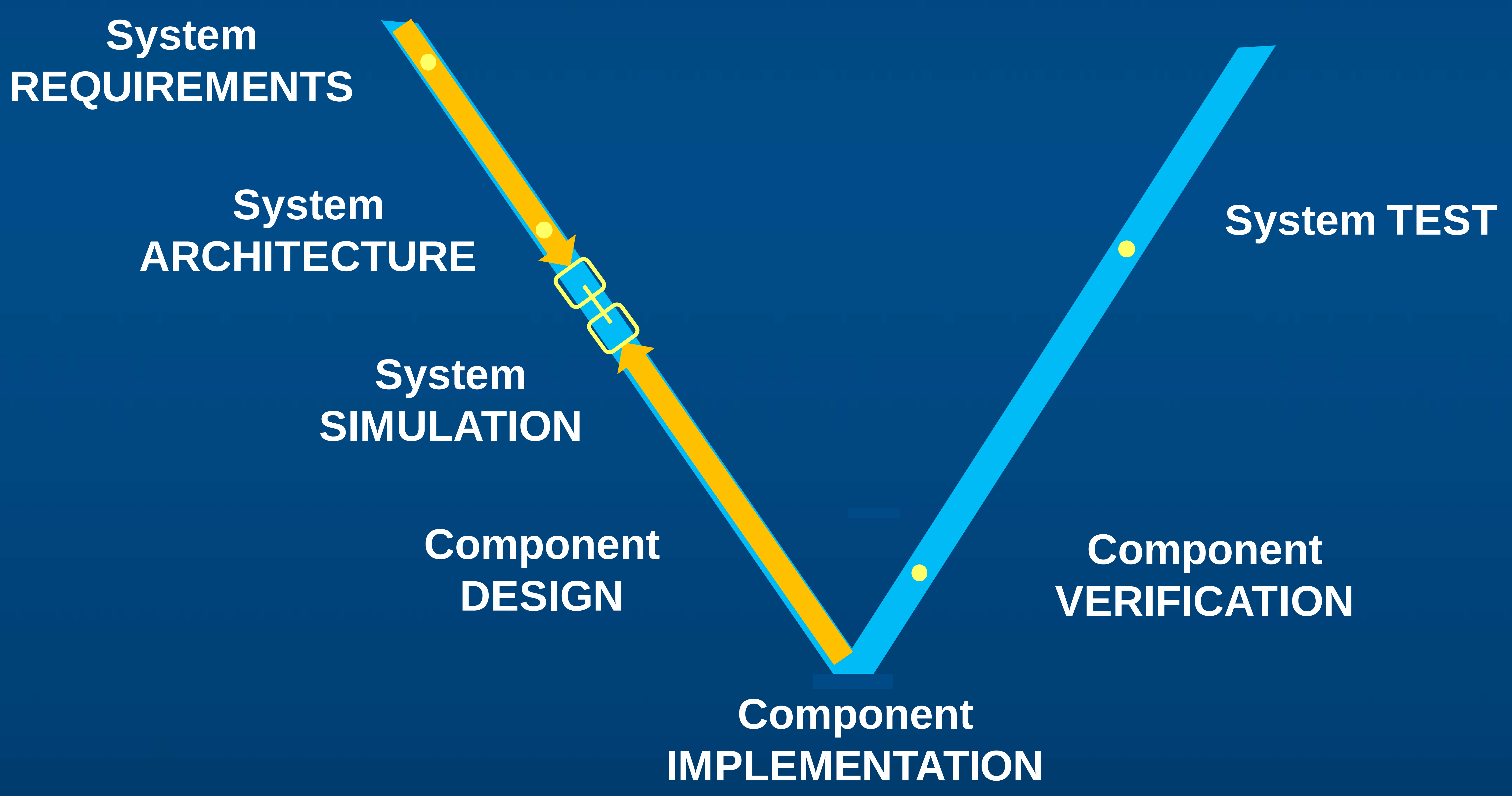




Components

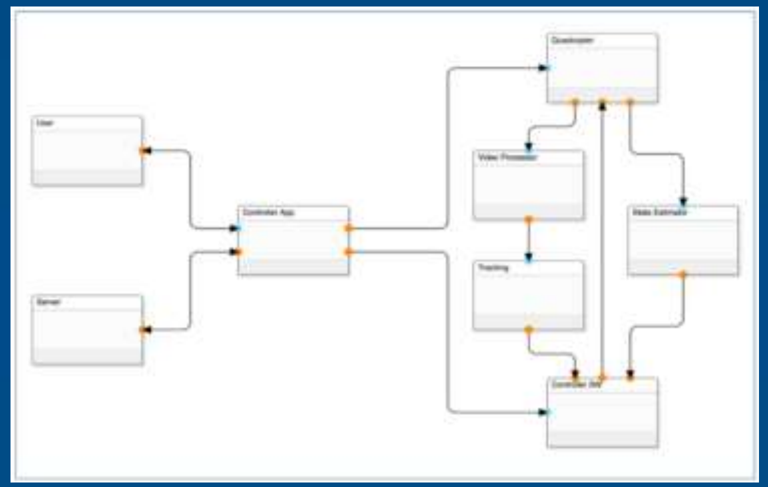


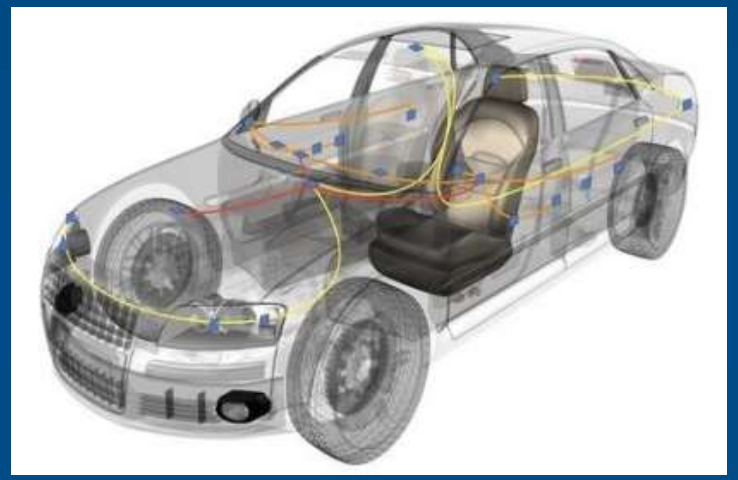
Linking top-down and bottom-up workflows



Types of models

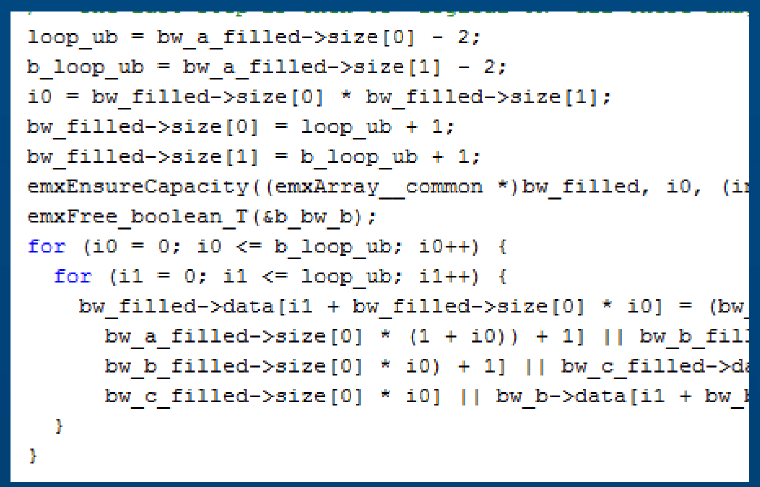
Systems





Software

Physics



Components

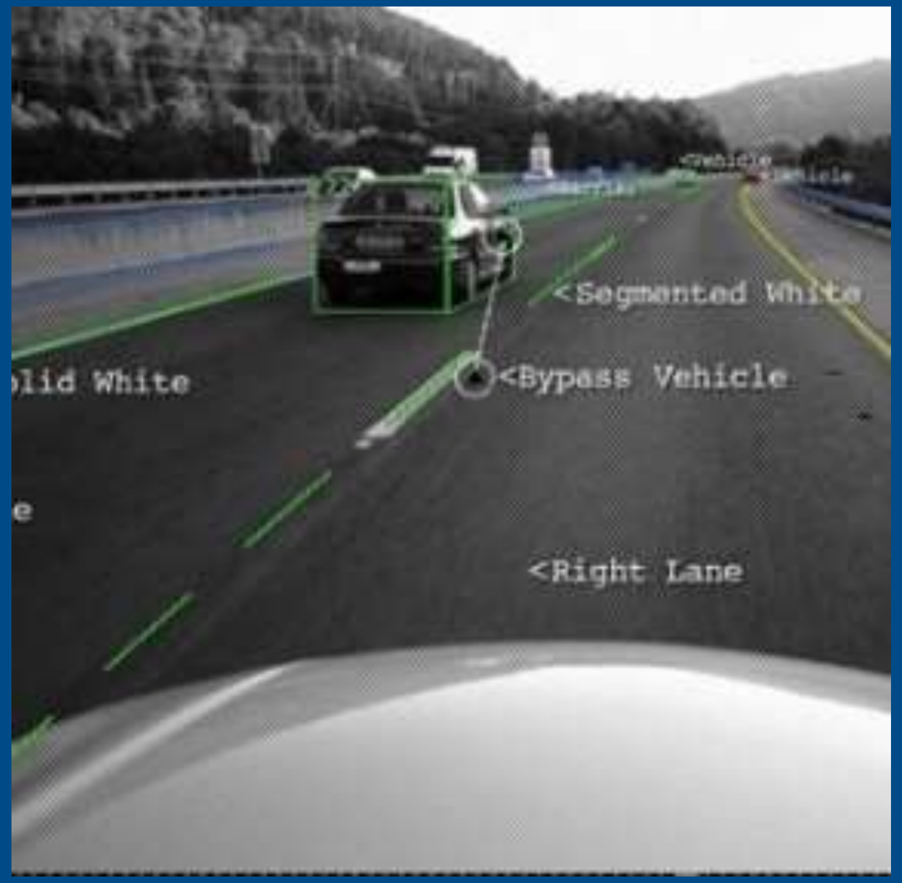
Deep solutions

Controls

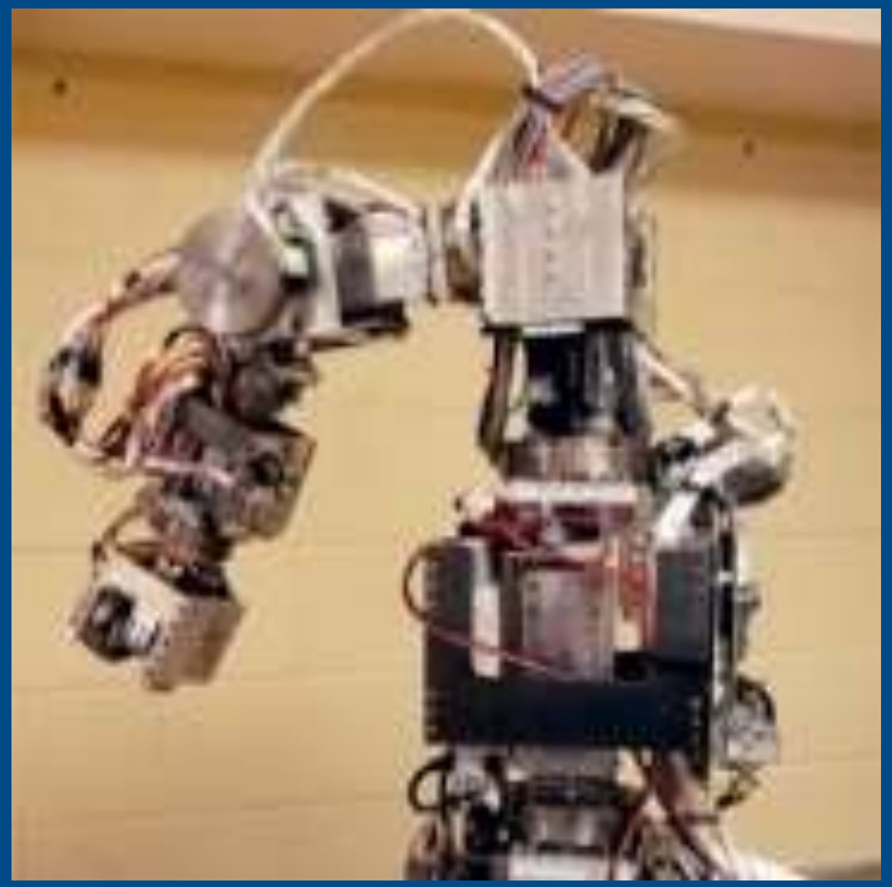
Signal Processing

Wireless

Vision



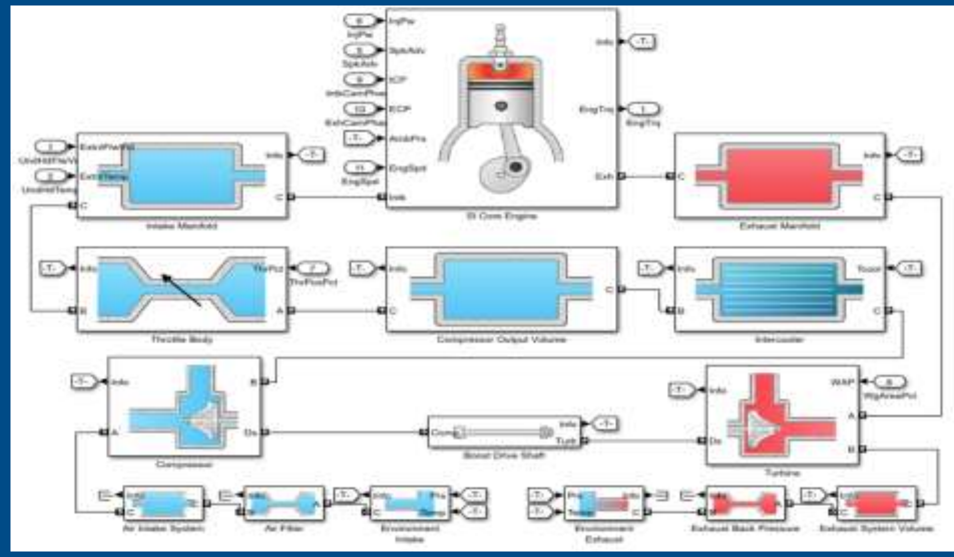
Robotics



Deep solutions

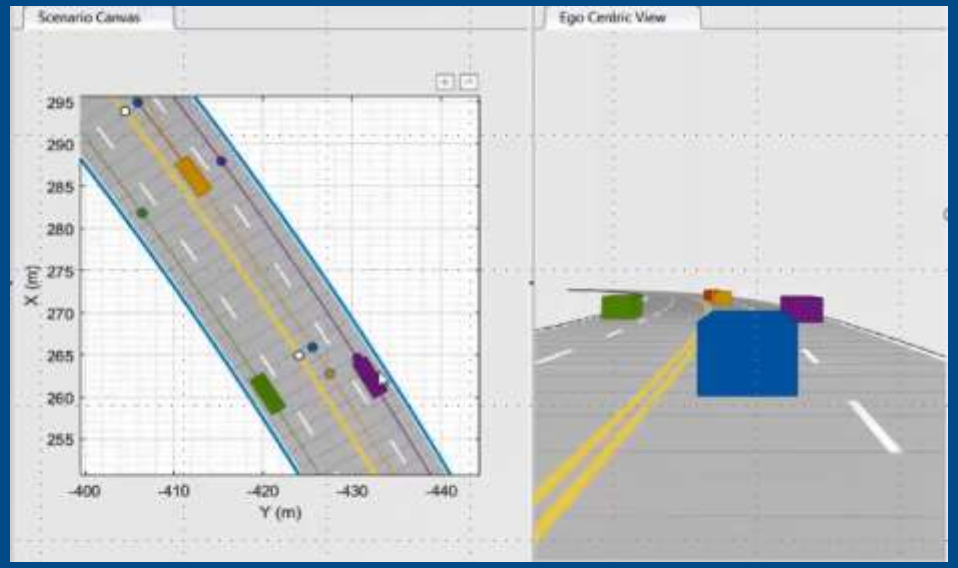


Automotive Products

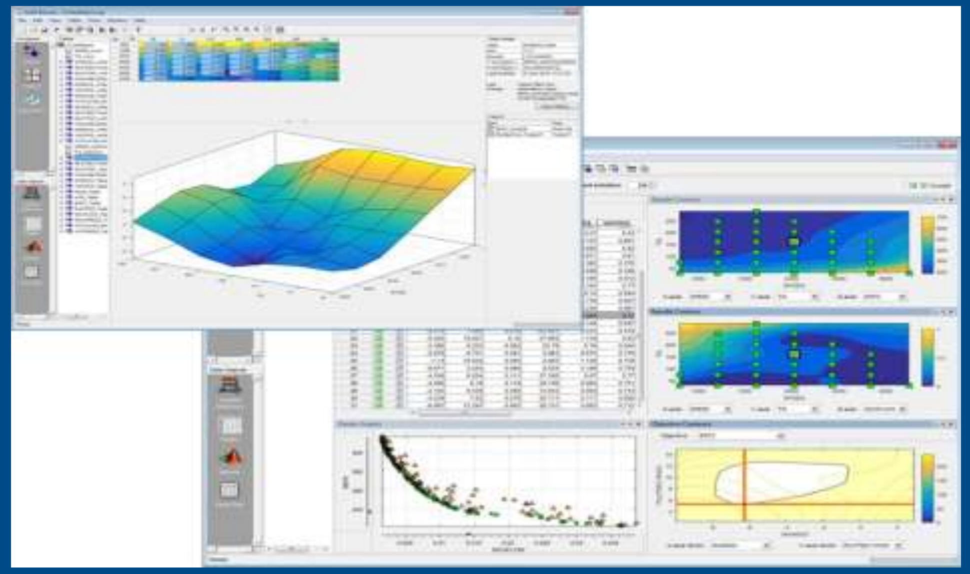


Powertrain

Vehicle

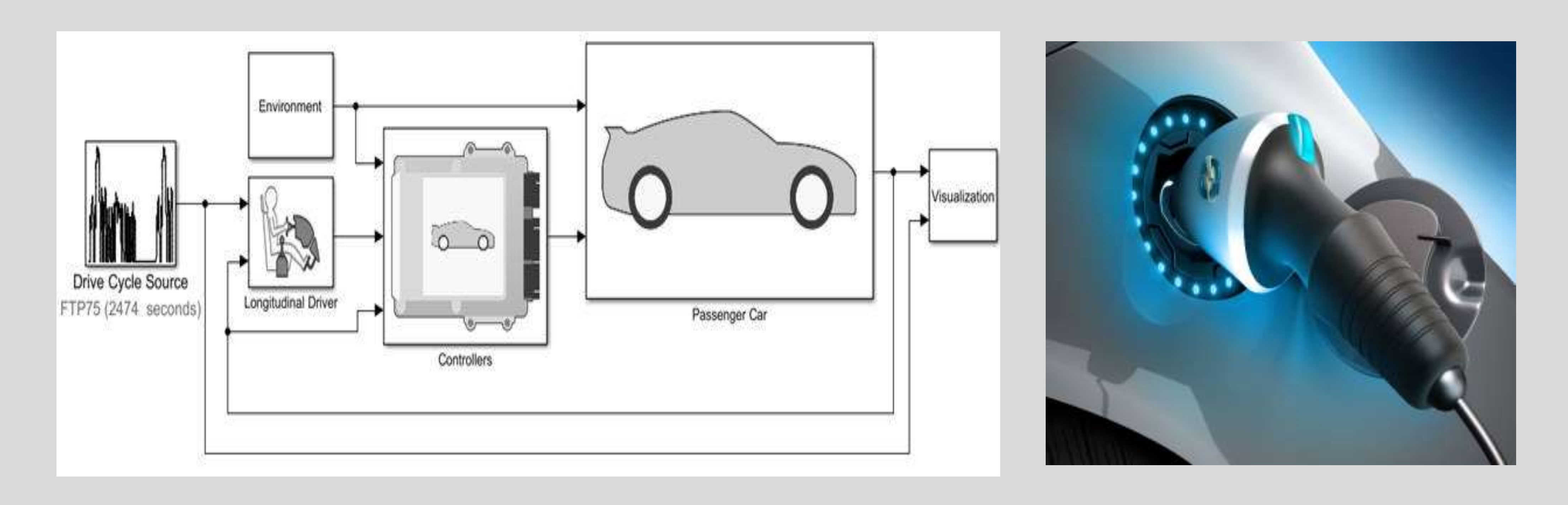


Automated Driving

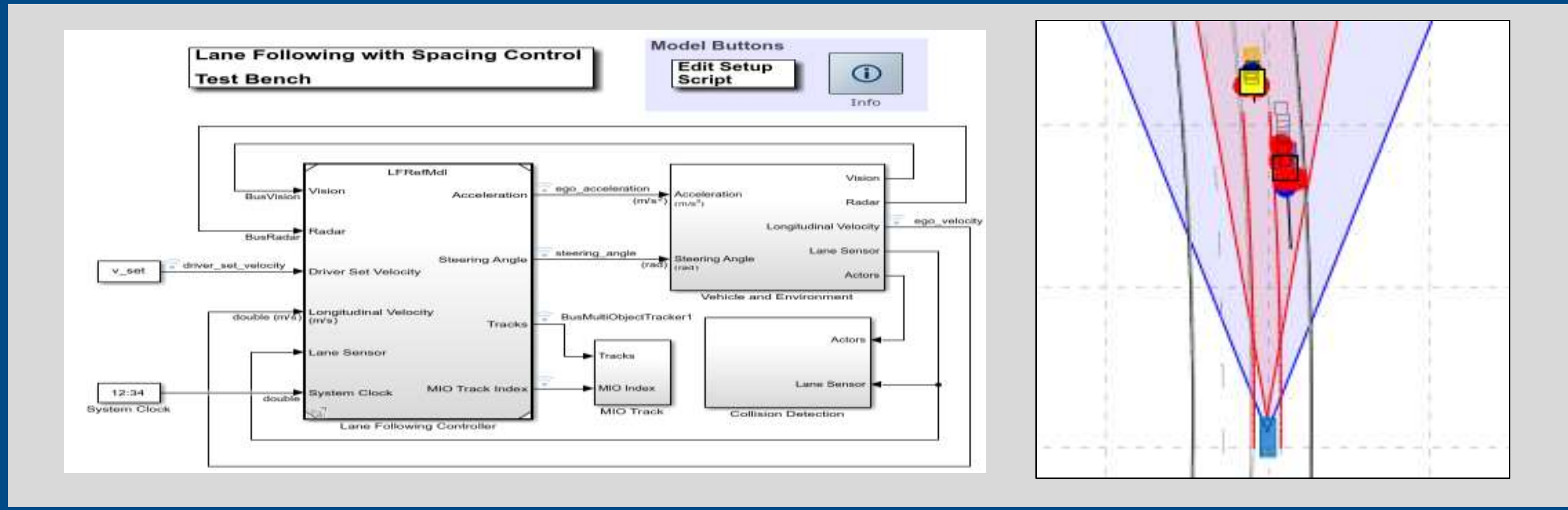


Calibration

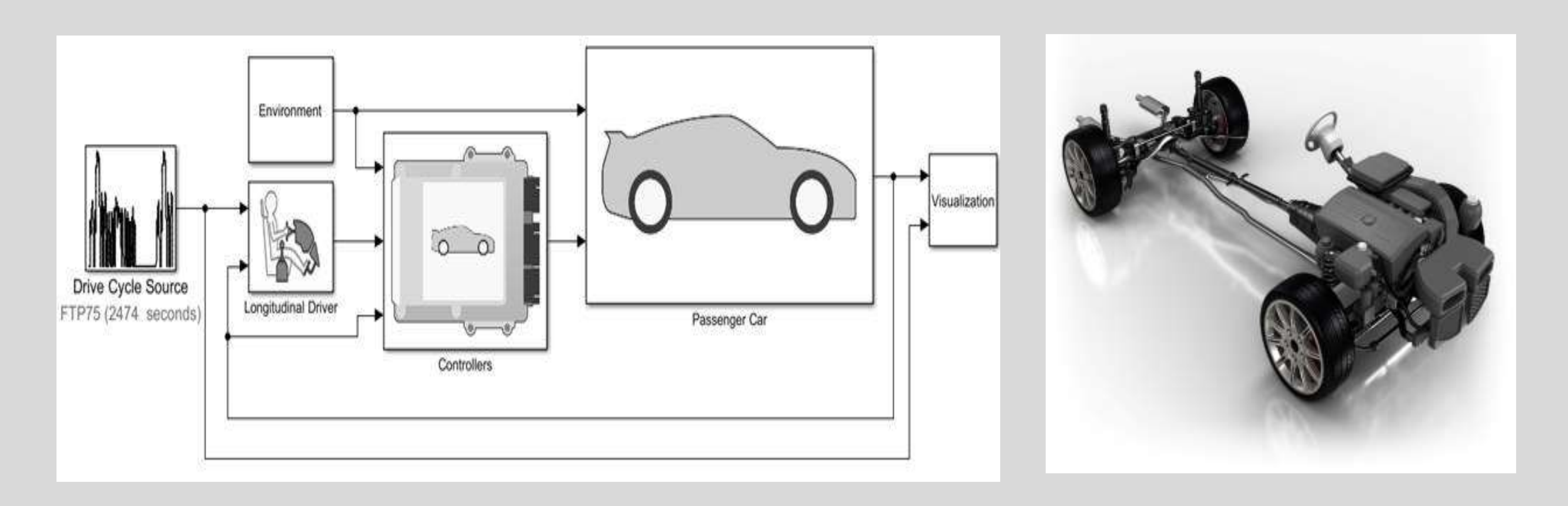
Automotive Reference Applications



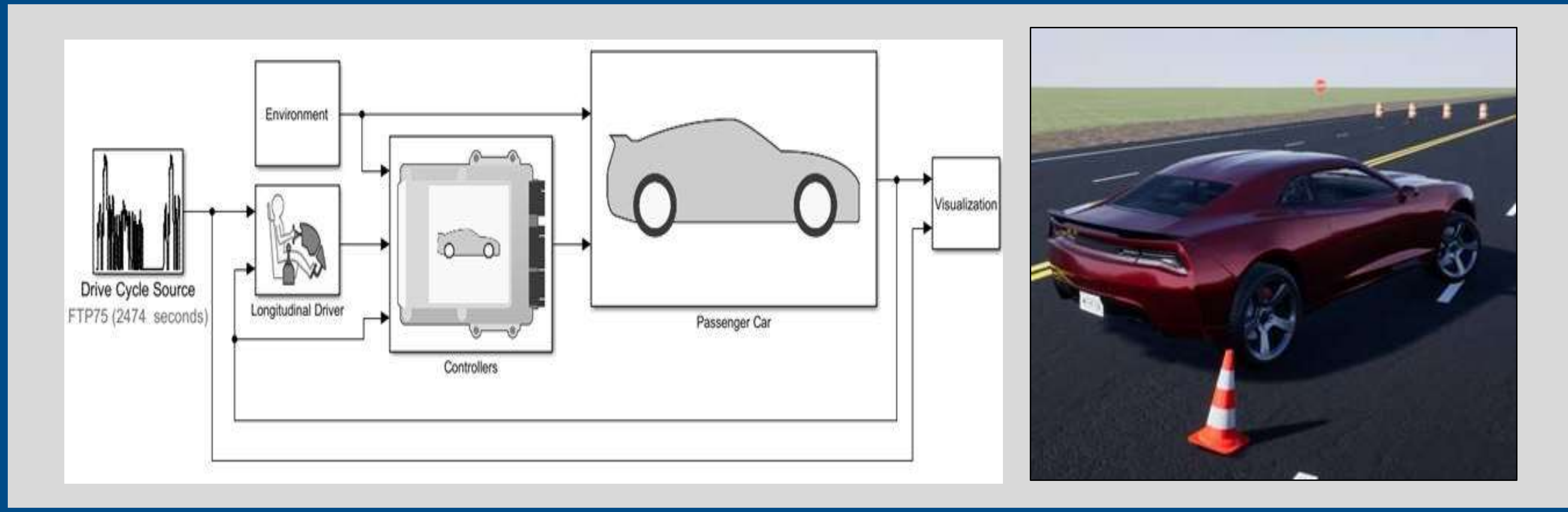
Pure EV



Lane Keeping Assist



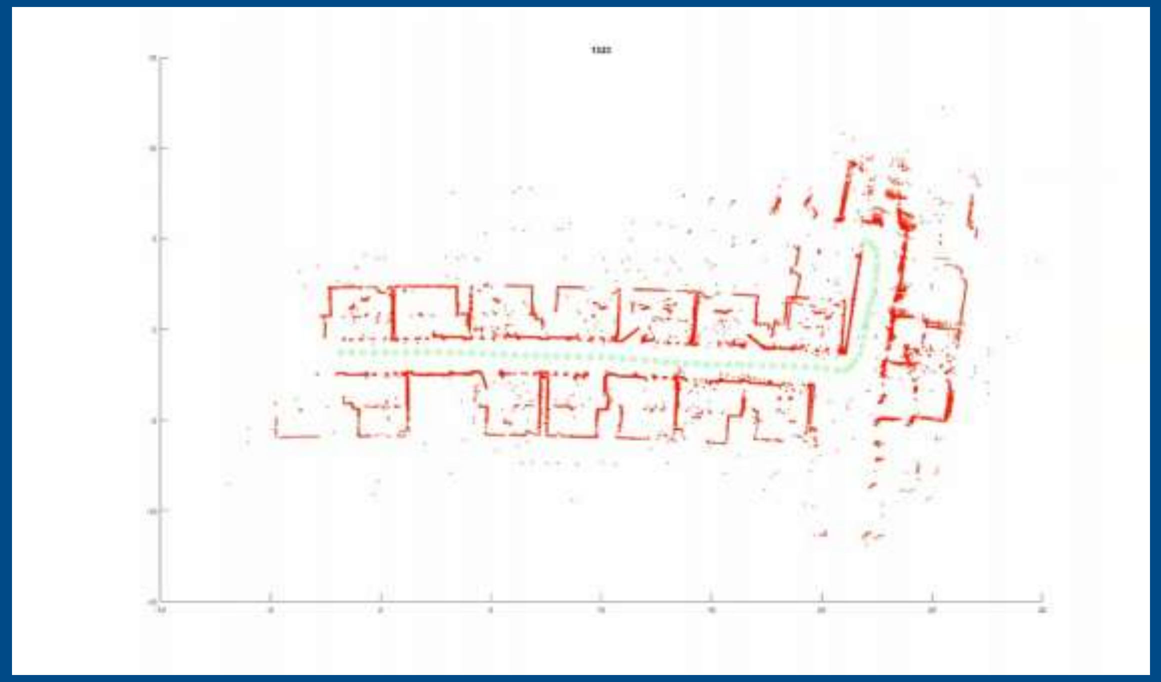
Hybrid Powertrain



Car Vehicle Dynamics

Deep solutions for autonomous systems

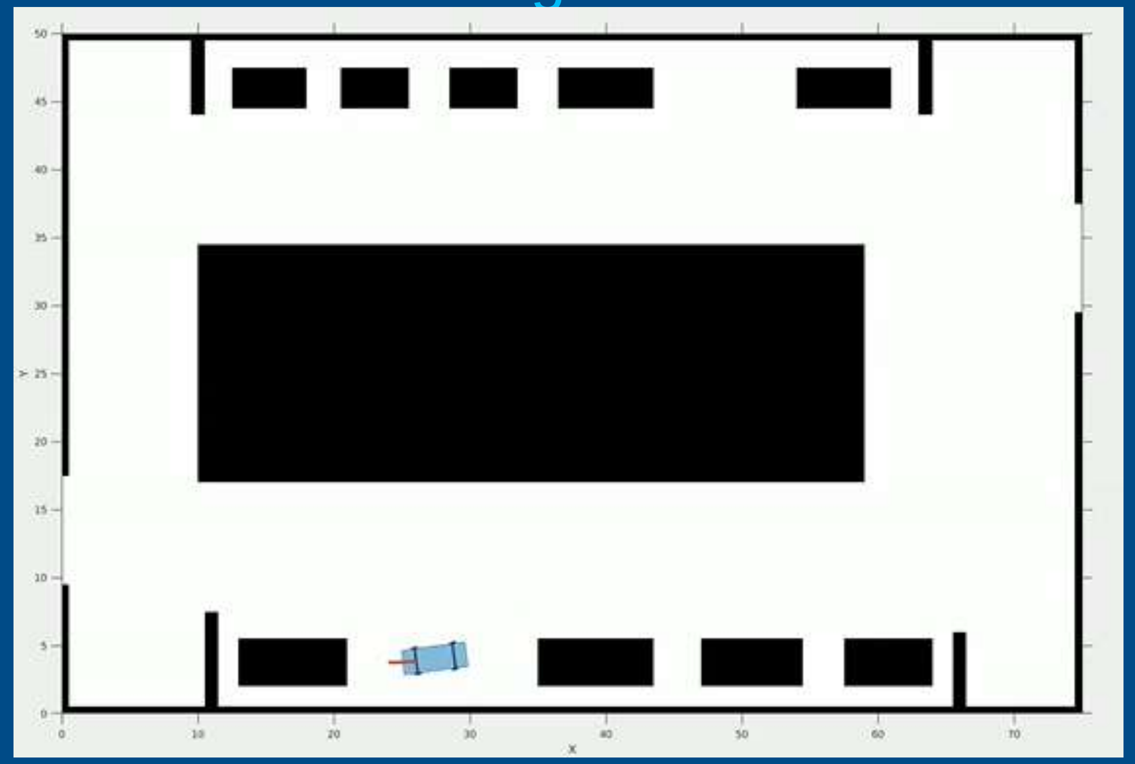
SLAM (18a)
Robotics System Toolbox



Semantic Segmentation (17b)
Automated Driving
System Toolbox

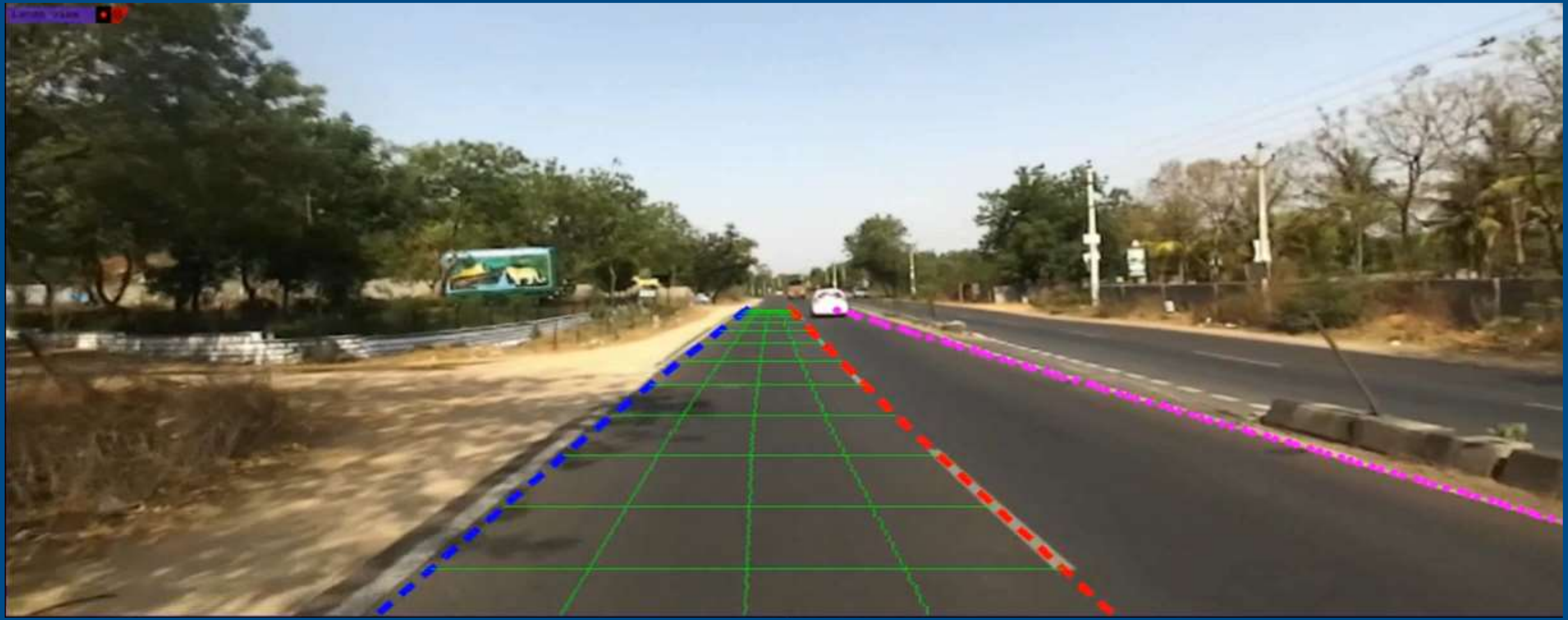
Localization → Planning → Control →  → Perception → Localization

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


AutoVrtlEnv (64-bit, PCD3D_SM5)
MTHFUN

MATLAB Workflow for Deep Learning:

Access Data

Preprocess

- MUNGING/LABELING
- FUSION
- DENOISING

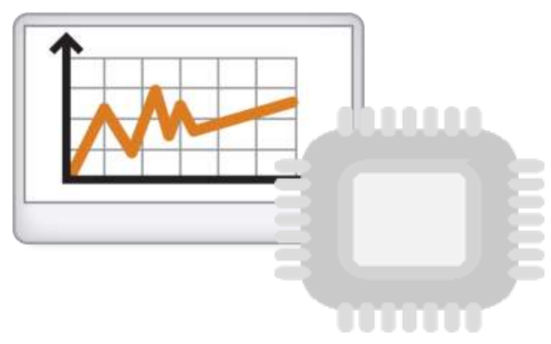
Access Models

- 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

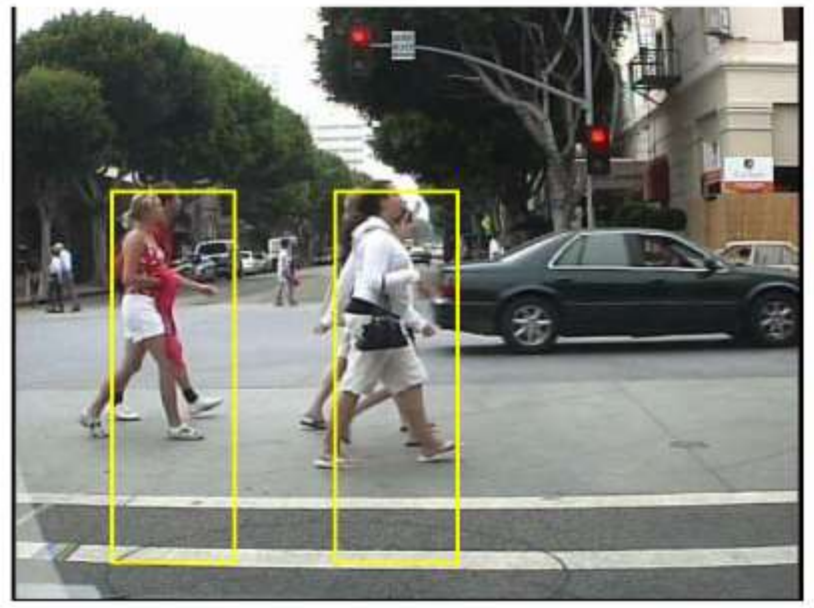
ARM

Artificial Intelligence for your applications

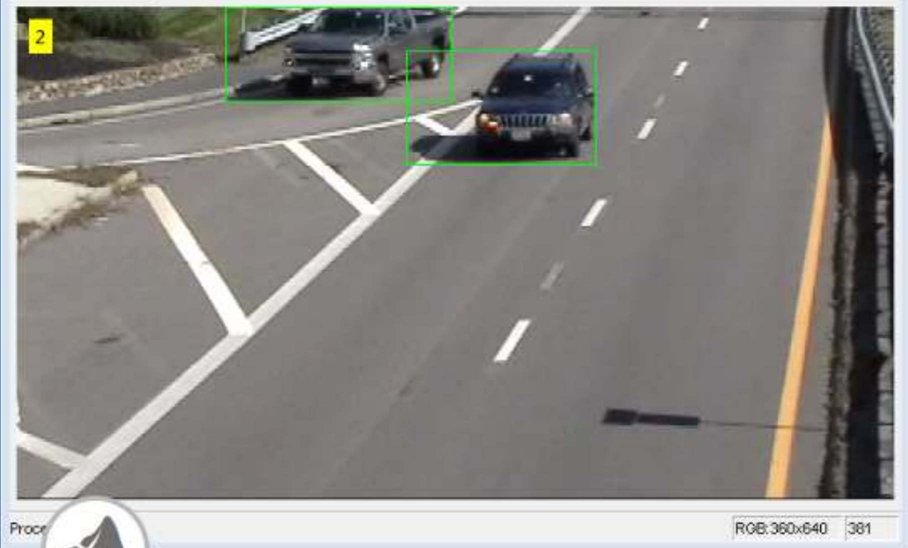
- Application examples

Object Detection Using Deep Learning

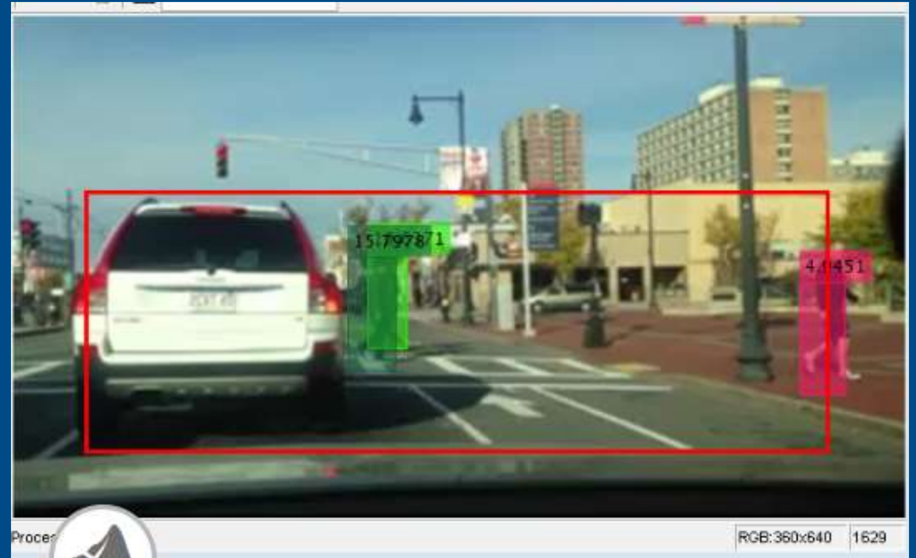
Traffic Sign Detection and Recognition



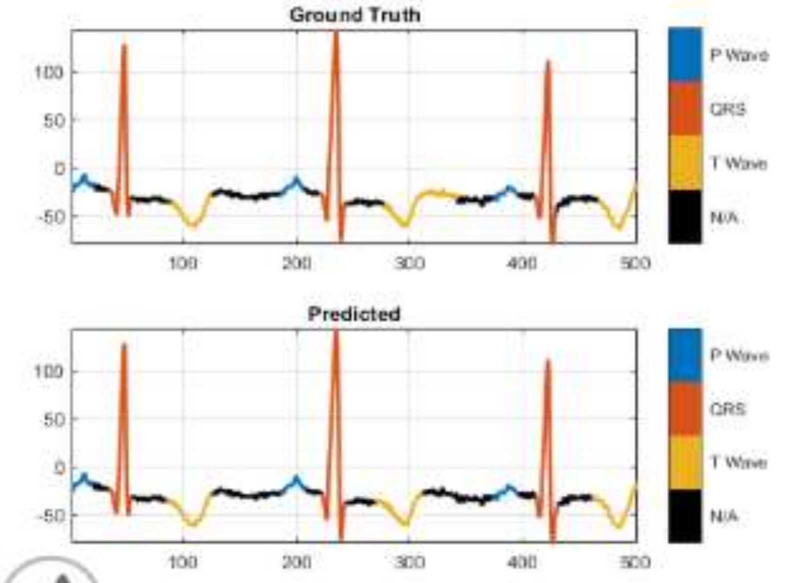
Pedestrian Detection



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

Reinforcement Learning Toolbox

Modeling

Simulation

+

Automation

Coding

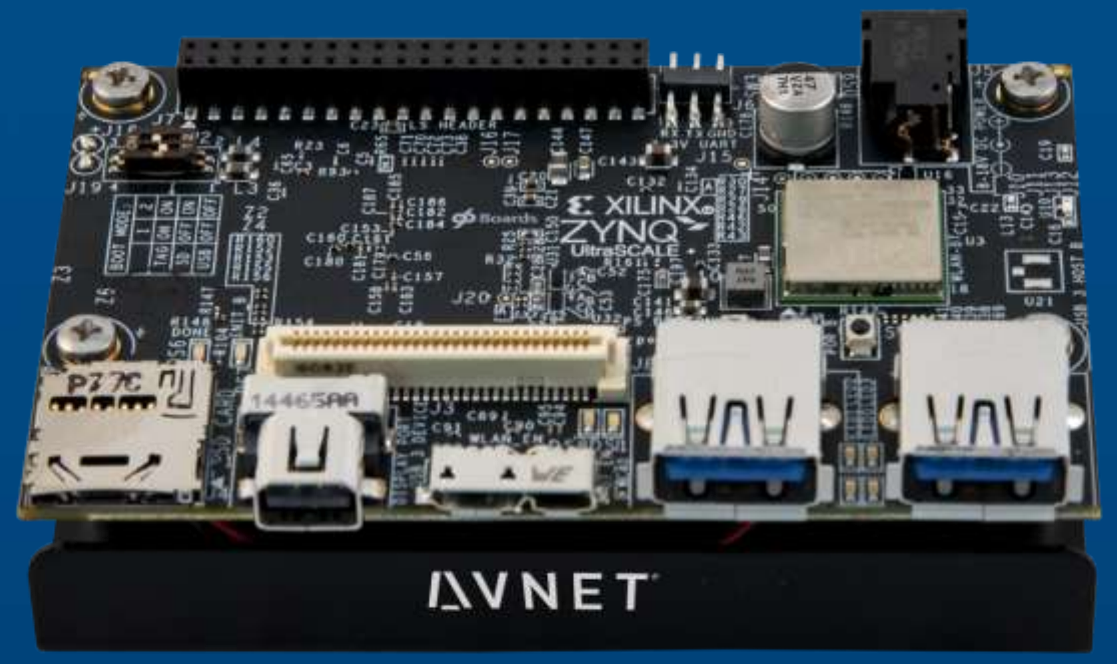
Coding

```
#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

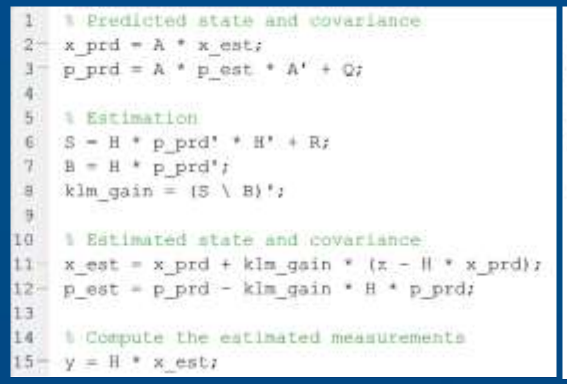
GPU
Fastest

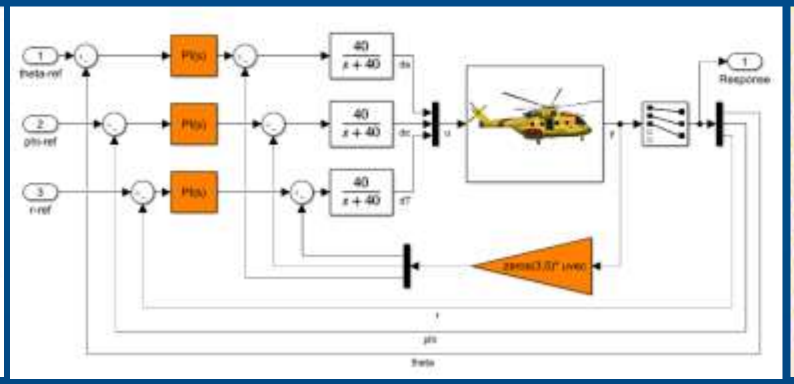
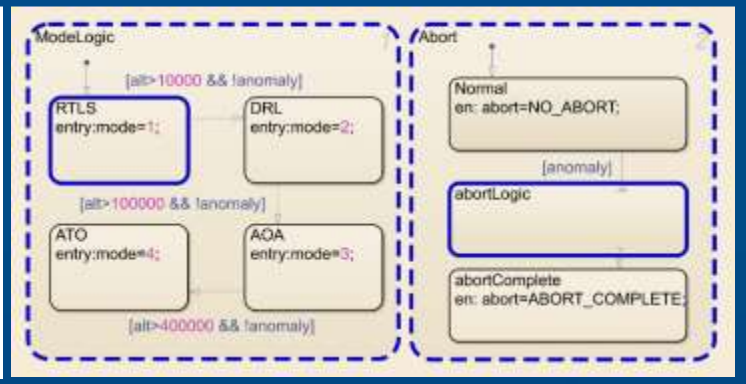
FPGA / ASIC
Lowest Power



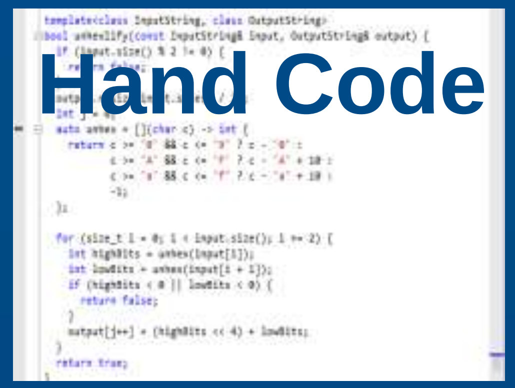
CPU
Low Cost

Model-Based Design   C/C++



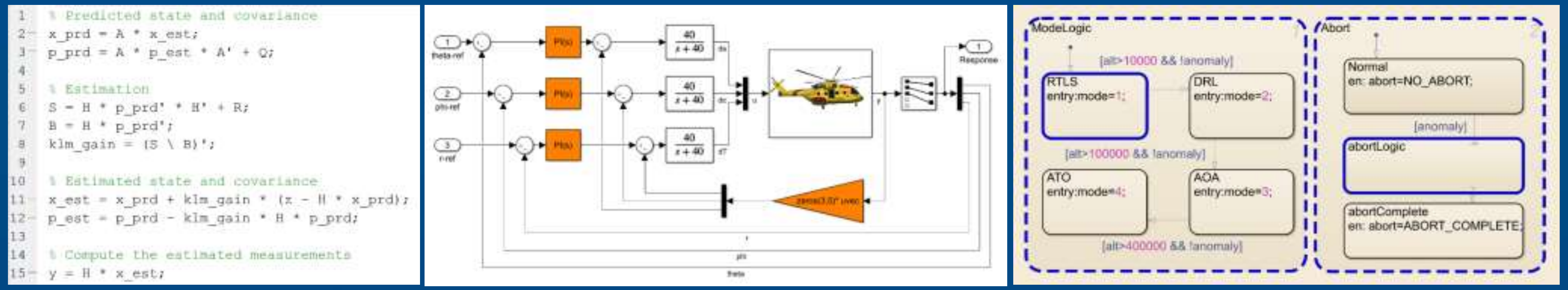
VS



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

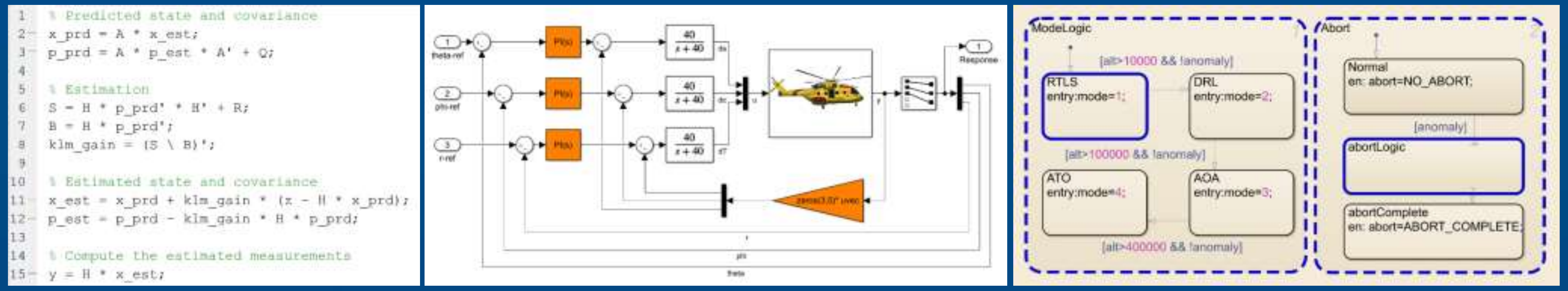
Model-Based Design

C/C++ Libraries



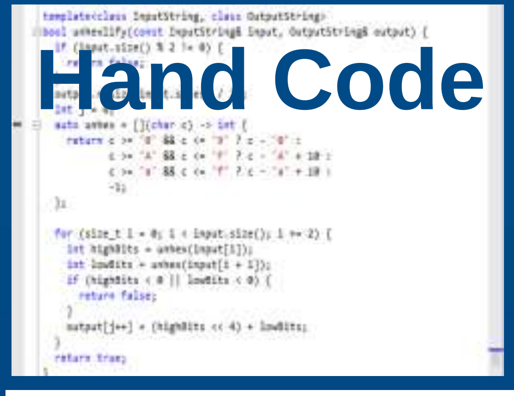
- No wrappers
- No data typing
- No data copies

Model-Based Design



- No wrappers
- No data typing
- No data copies

C/C++ Libraries



Hand Code

Internal Libraries

Vendor Libraries

Middleware

ROS AUTOSAR DDS

Modeling

Simulation

+

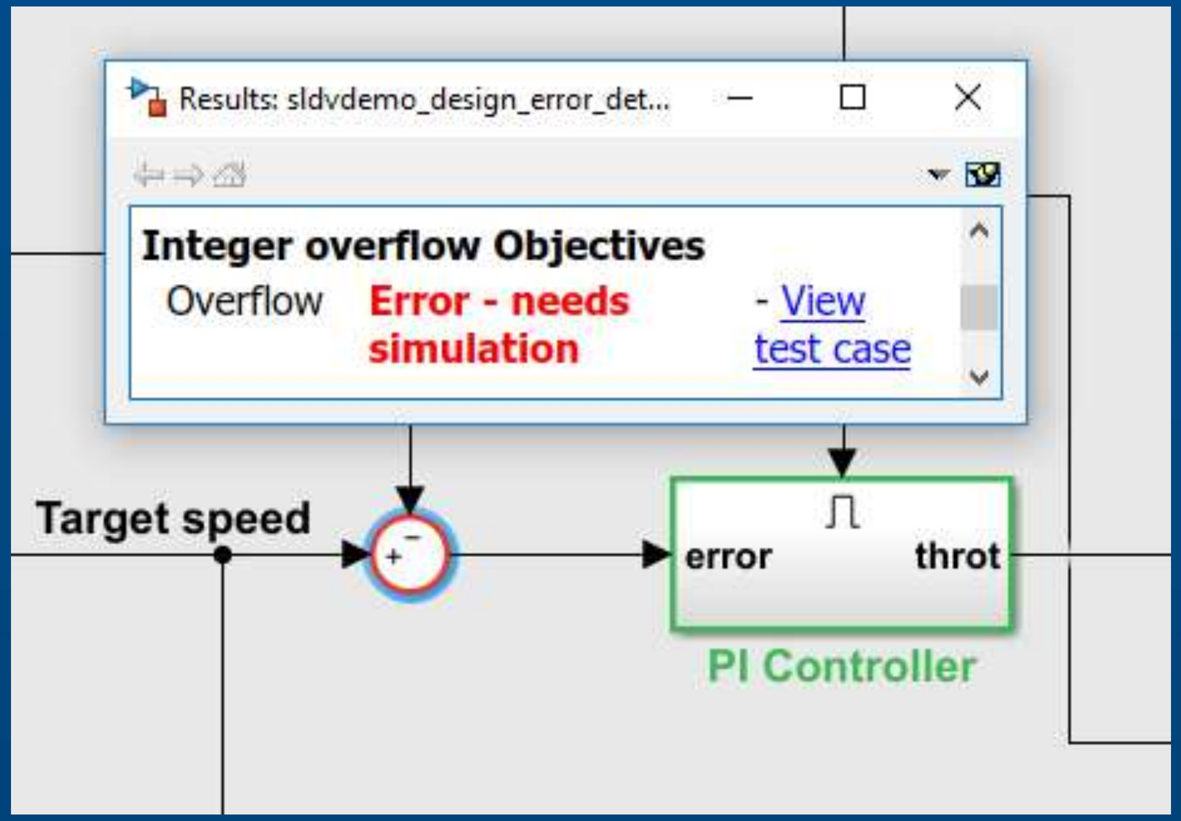
Automation

Coding

Verification

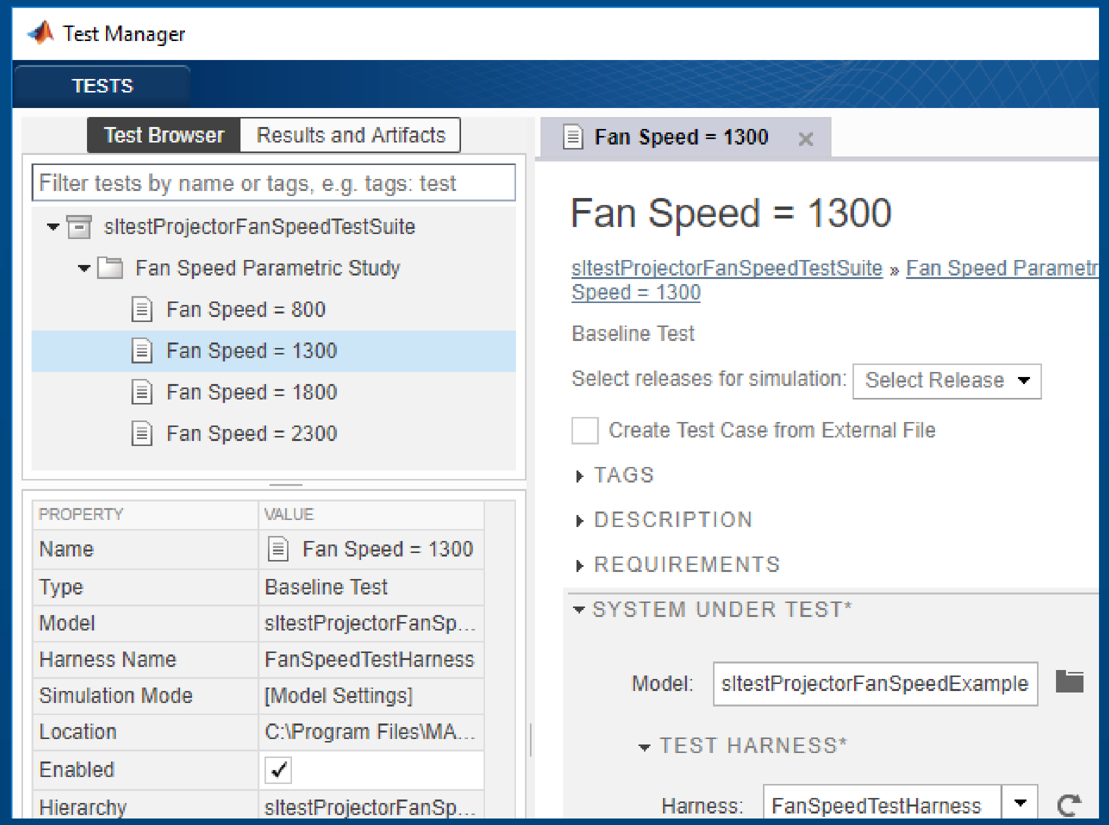
Automated Test and Verification

Find bugs



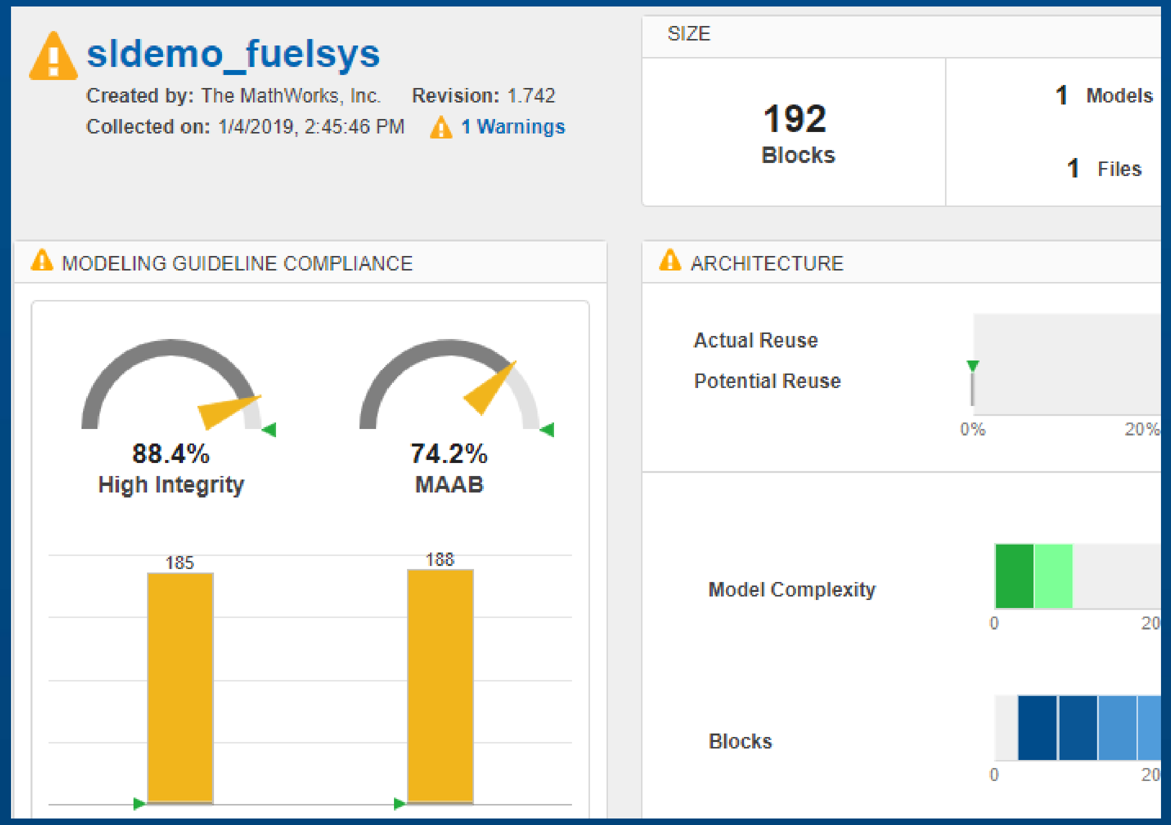
Simulink Design Verifier
Polyspace Bug Finder

Manage tests



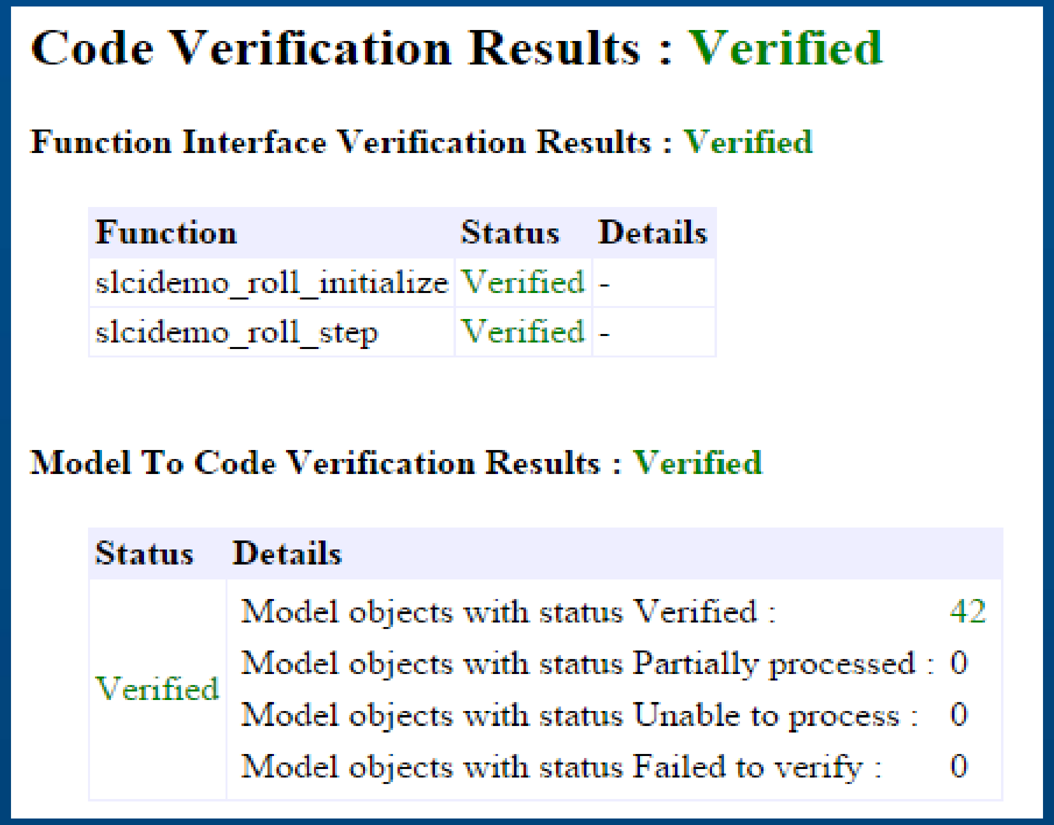
Simulink Test

Check & Coverage



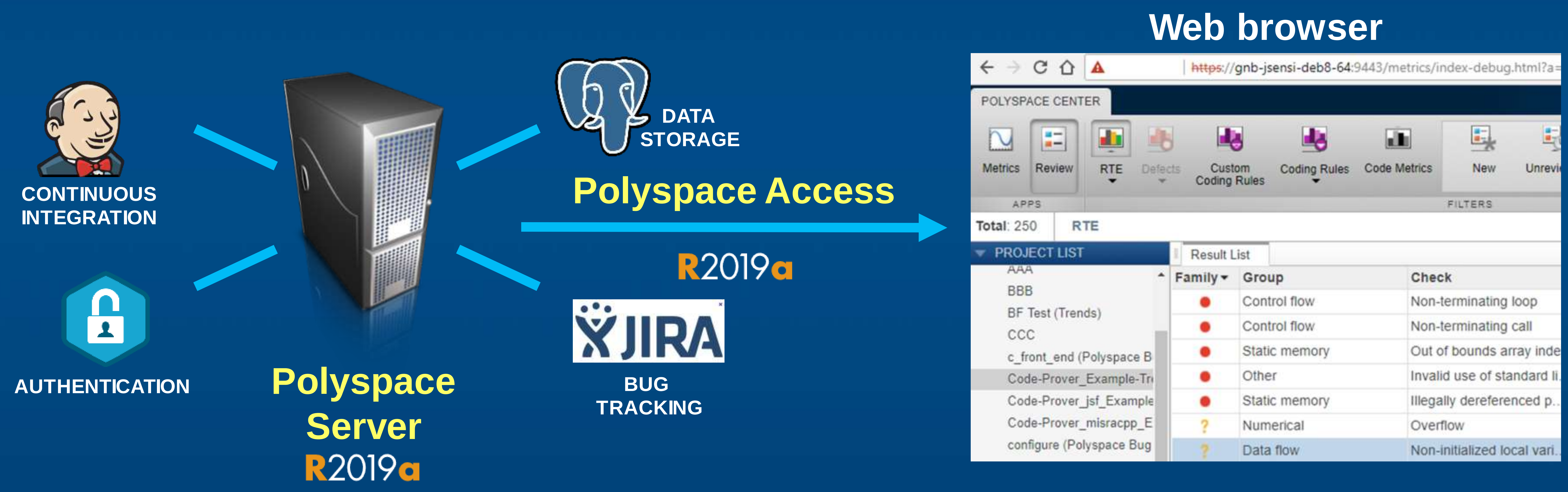
Simulink Check
Simuink Coverage

Inspect code



Simulink Code Inspector

Online Access for Test and Verification



Model-Based Design

Systematic use of models throughout the development process

Modeling

Simulation

+

Automation

Coding | **Verification**

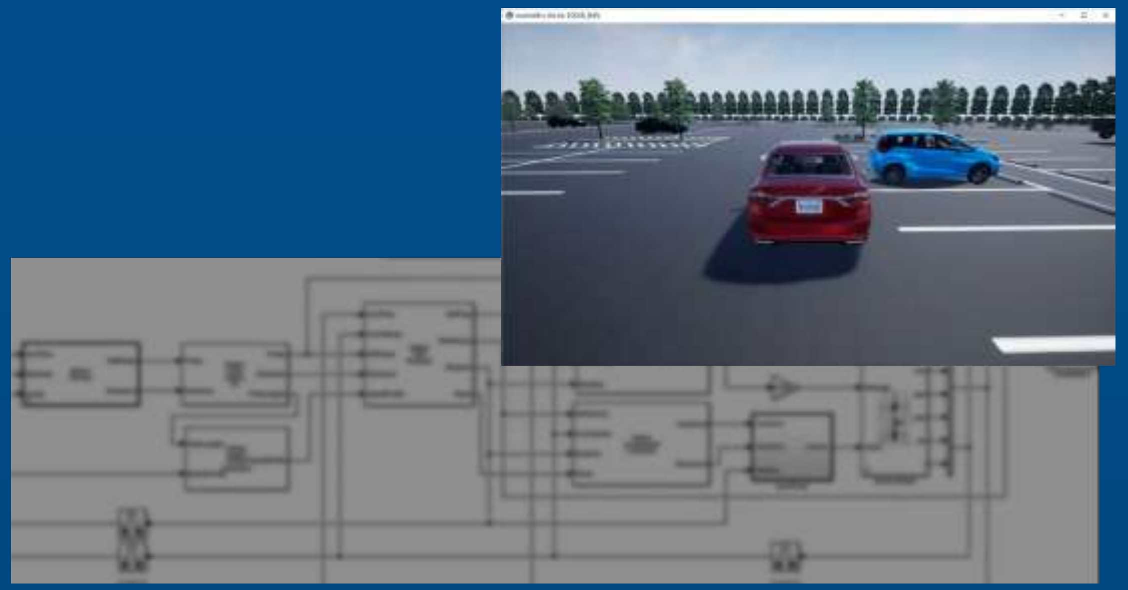
Fast repeatable tests

Fast agile development loops

Who will be successful in the future?

Mechanical-centric

Model-centric



Software-centric

Comprehensive models
Simulation based testing
Generate code and automate verification

Enjoy the conference