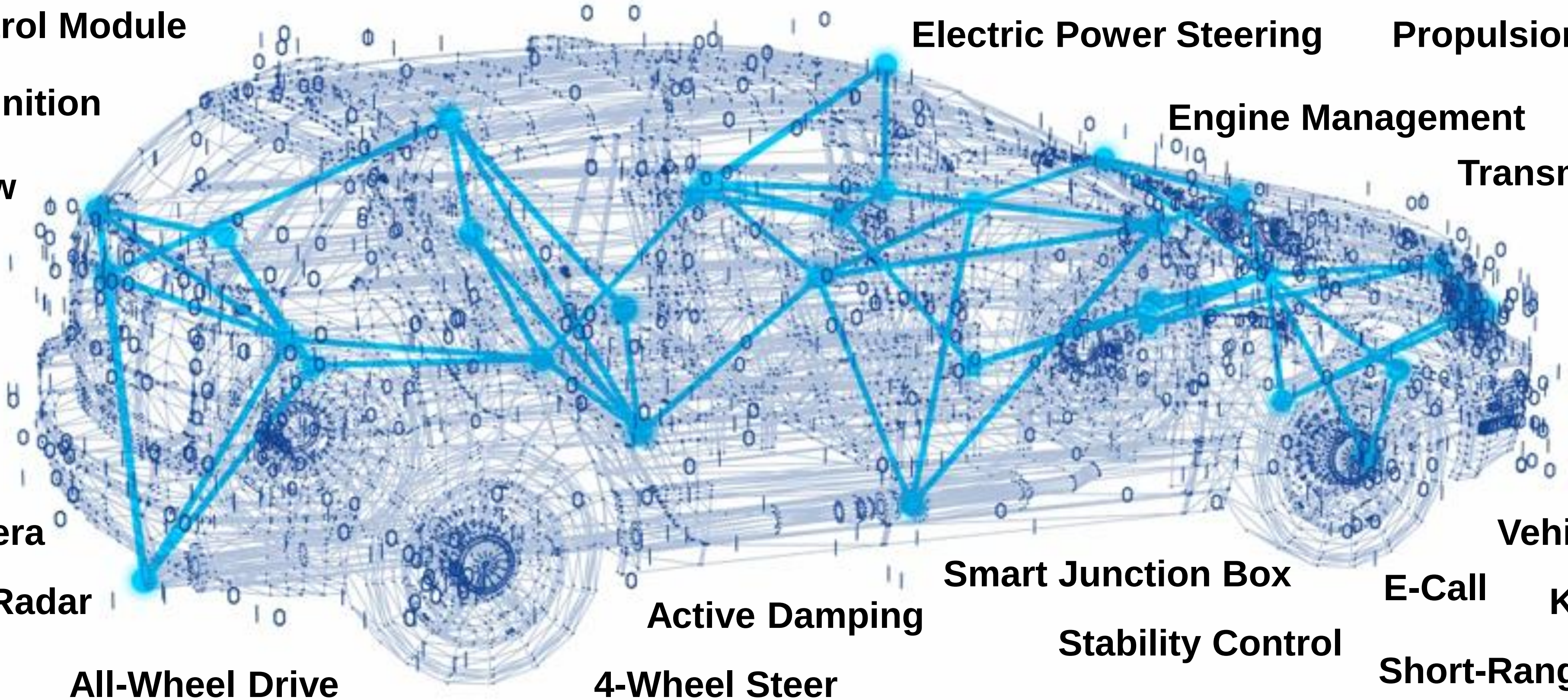
MathWorks Vision for Systematic Verification and Validation

Bill Aldrich

Senior Development Manager

Simulink Verification and Validation, Simulink Design Verifier

Growing Complexity of Automotive Controls



Emergency Braking

Body Control Module

Voice Recognition

Power Window

Vehicle-to-
Infrastructure

Power Liftgate

Power Seat

Back-up Camera

Long-Range Radar

Stability Control

Adaptive Cruise Control

Automatic Parking

Smart Junction Box

Battery Management

Instrument Panel

Airbag

Electric Power Steering

Infotainment

DC/DC Converter

Propulsion Motor Control

Engine Management

Navigation

Transmission Control

Forward Camera

Adaptive Front
Lighting

HVAC Control

Vehicle-to-Vehicle

Keyless Entry

E-Call

Short-Range Radar

Smart Junction Box

Stability Control

Active Damping

4-Wheel Steer

All-Wheel Drive

Tire Pressure Monitor

Ultrasonic Sensor

Source of graphic: http://360.here.com/2013/11/28/putting-firmly-drivers-seat/

Growing Complexity of Automotive Controls

Emergency Braking

Stability Control

Automatic Parking

Battery Management

Instrument Panel

Infotainment

Adaptive Cruise Control

Smart Junction Box

Airbag

DC/DC Converter

Body Control Module

Propulsion Motor Control

Voice Recognition

anagement

Navigation

Power Window

Transmission Control

Vehicle-to-Infrastructure

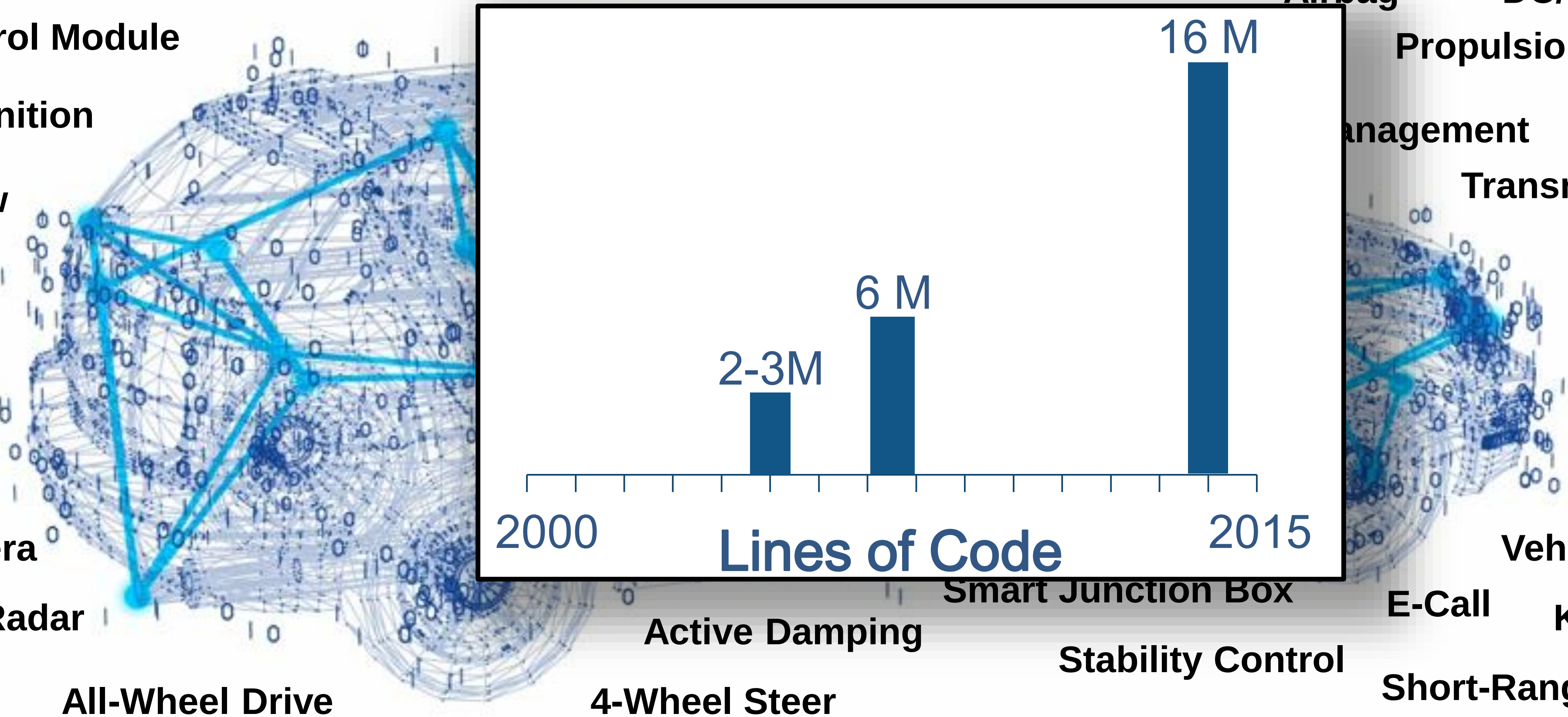
Forward Camera

Adaptive Front Lighting

Power Liftgate

HVAC Control

Power Seat



Back-up Camera

Vehicle-to-Vehicle

Smart Junction Box

E-Call

Long-Range Radar

Active Damping

Keyless Entry

Stability Control

Short-Range Radar

All-Wheel Drive

4-Wheel Steer

Siemens, “Ford Motor Company Case Study,” Siemens PLM Software, 2014
McKendrick, J. “Cars become ‘datacenters on wheels’, carmakers become software companies,” ZDJNet, 2013

Growing Complexity of Automotive Controls

Stability Control

Automatic Parking

Battery Management

Infotainment

Emergency Braking

Instrument Panel

Adaptive Cruise Control

Smart Junction Box

Airbag

DC/DC Converter

Body Con

Propulsion Motor Control

Voice Reco

Navigation

anagement

Power Windo

Transmission Control

Vehicle-to-Infrastructure

Forward Camera

Adaptive Front Lighting

Power Liftgate

HVAC Control

Power Seat

Back-up Camera

Vehicle-to-Vehicle

Smart Junction Box

E-Call

Long-Range Radar

Active Damping

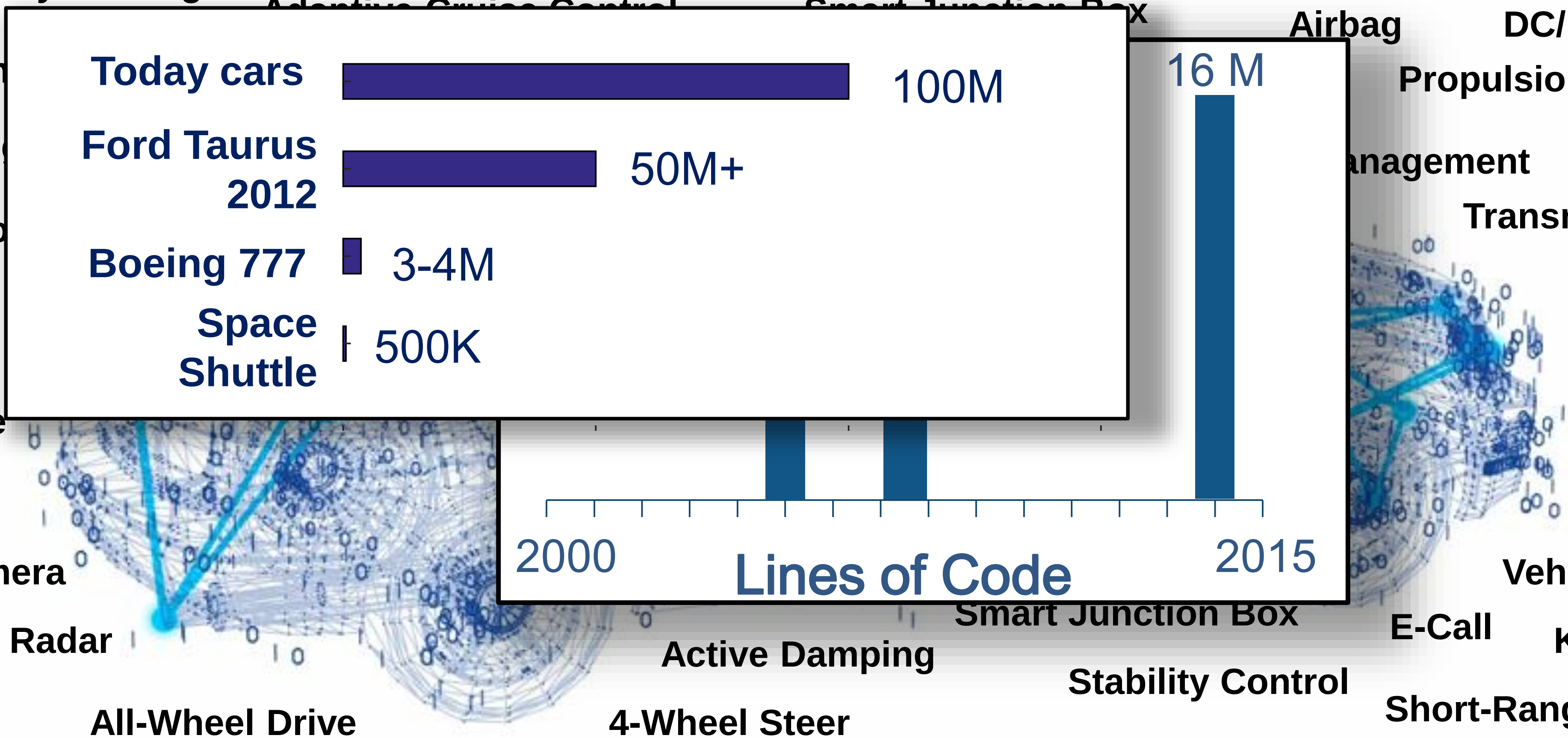
Keyless Entry

Stability Control

Short-Range Radar

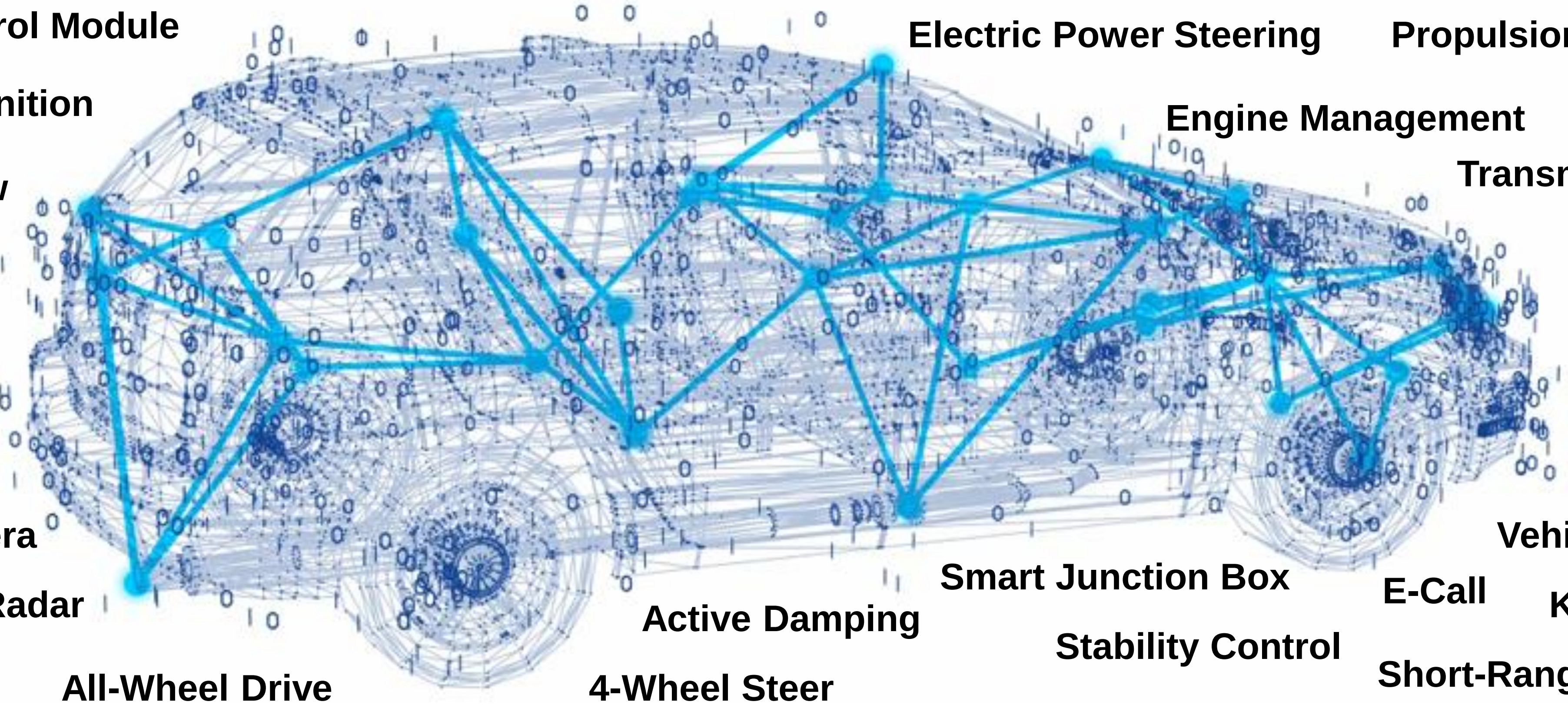
All-Wheel Drive

4-Wheel Steer



Source: https://interact.gsa.gov/sites/default/files/J3061%20JP%20presentation.pdf

Growing Complexity of Automotive Controls



Stability Control

Automatic Parking

Battery Management

Infotainment

Emergency Braking

Instrument Panel

Adaptive Cruise Control

Smart Junction Box

Airbag

DC/DC Converter

Body Control Module

Electric Power Steering

Propulsion Motor Control

Voice Recognition

Engine Management

Navigation

Power Window

Transmission Control

Vehicle-to-Infrastructure

Forward Camera

Adaptive Front Lighting

Power Liftgate

Power Seat

HVAC Control

Back-up Camera

Vehicle-to-Vehicle

Smart Junction Box

E-Call

Long-Range Radar

Keyless Entry

Active Damping

Stability Control

Short-Range Radar

All-Wheel Drive

4-Wheel Steer

Tire Pressure Monitor

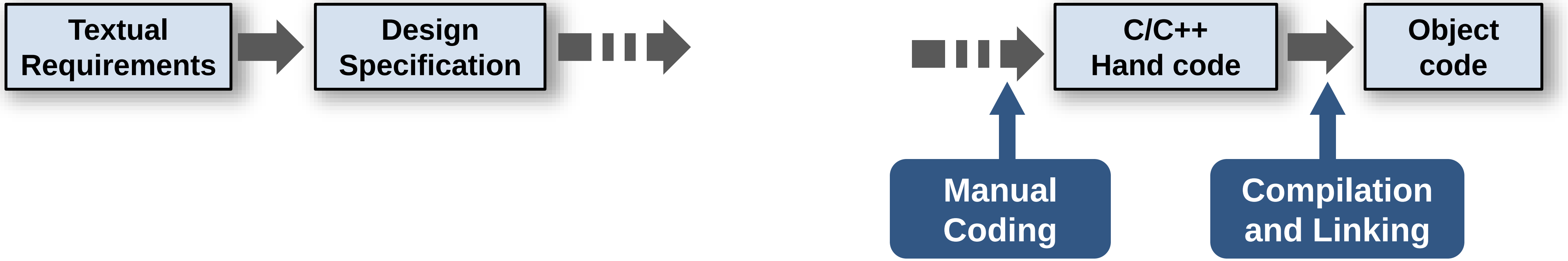
Ultrasonic Sensor

Source of graphic: http://360.here.com/2013/11/28/putting-firmly-drivers-seat/

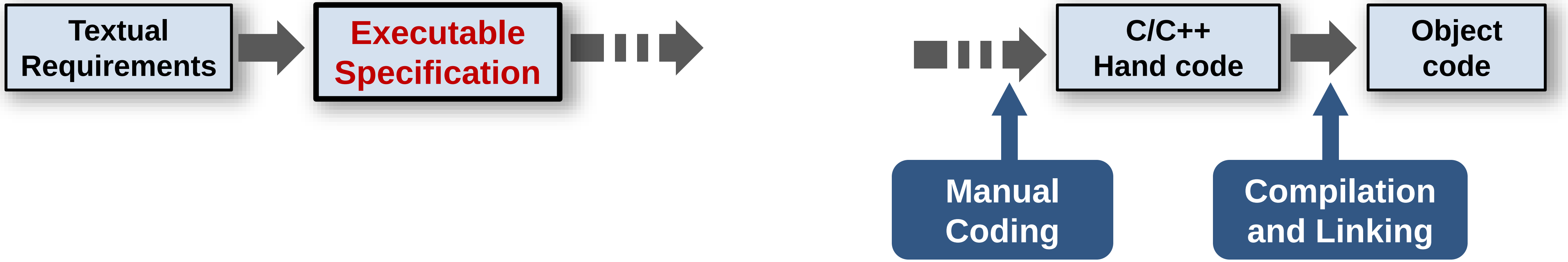
Development Challenges

- Representing complex systems
- Coordinating work across teams
- Working efficiently
- Ensuring quality

Traditional Development Process



Models for Specification



Textual Requirements

Executable Specification

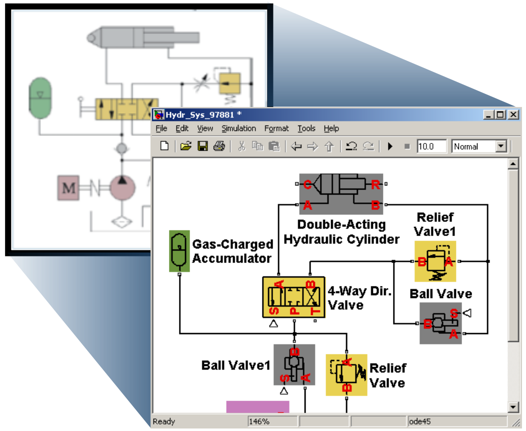
C/C++ Hand code

Object code

Manual Coding

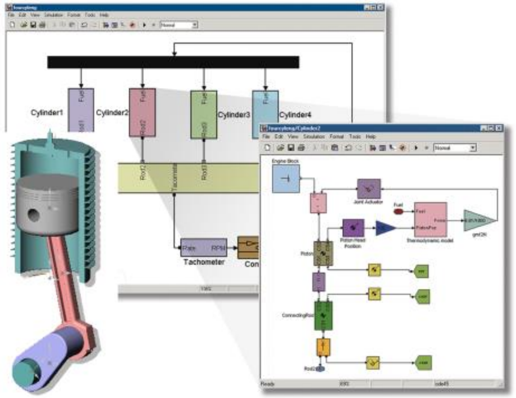
Compilation and Linking

Model Abstraction – Work at an appropriate level of detail

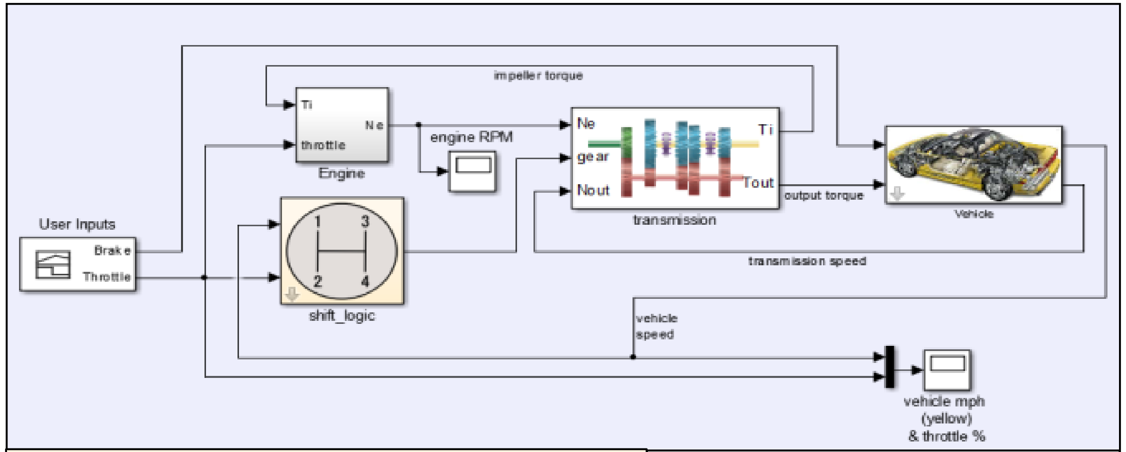


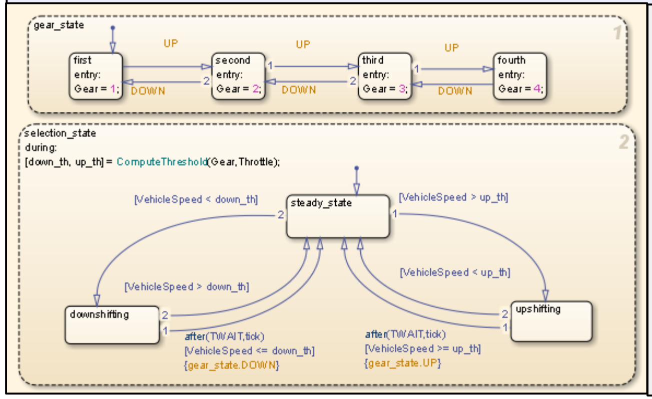
Simscape Fluids

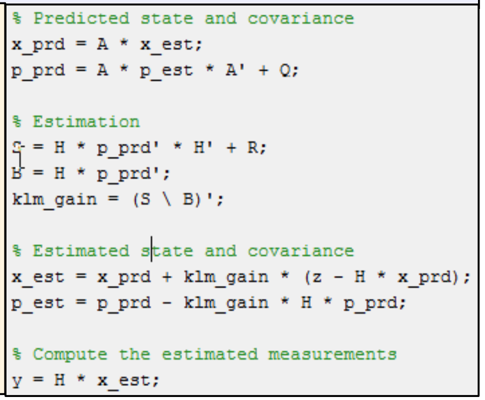
Simscape Multibody



Simulink





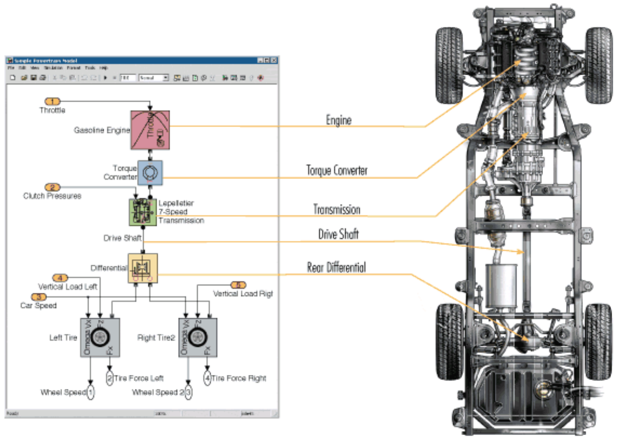


```
% Predicted state and covariance
x_prd = A * x_est;
p_prd = A * p_est * A' + Q;

% 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;
```

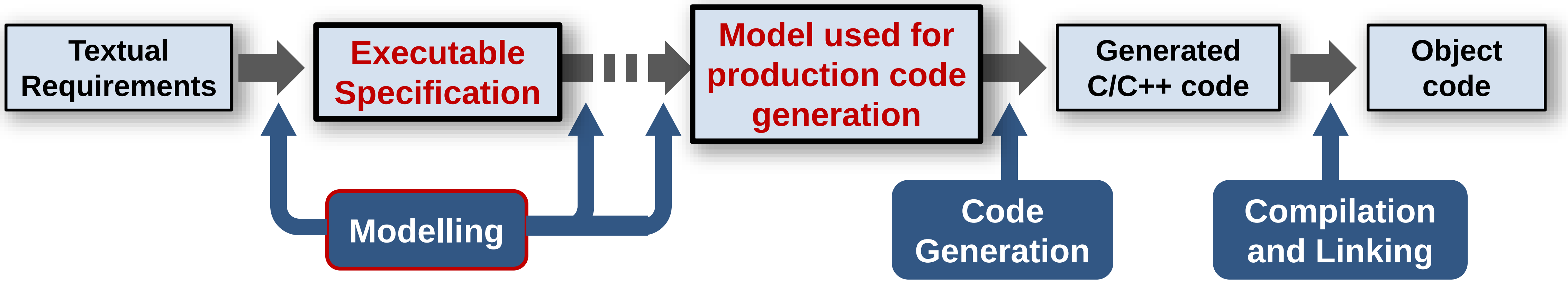
Stateflow

MATLAB

Simscape Driveline



Complete Model Based Design Workflow, Concept to Code



Textual Requirements

Executable Specification

Model used for production code generation

Generated C/C++ code

Object code

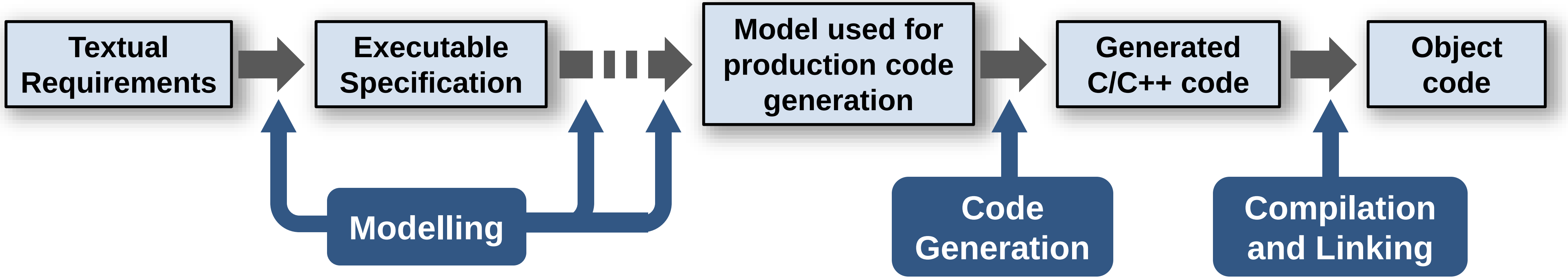
Modelling

Code Generation

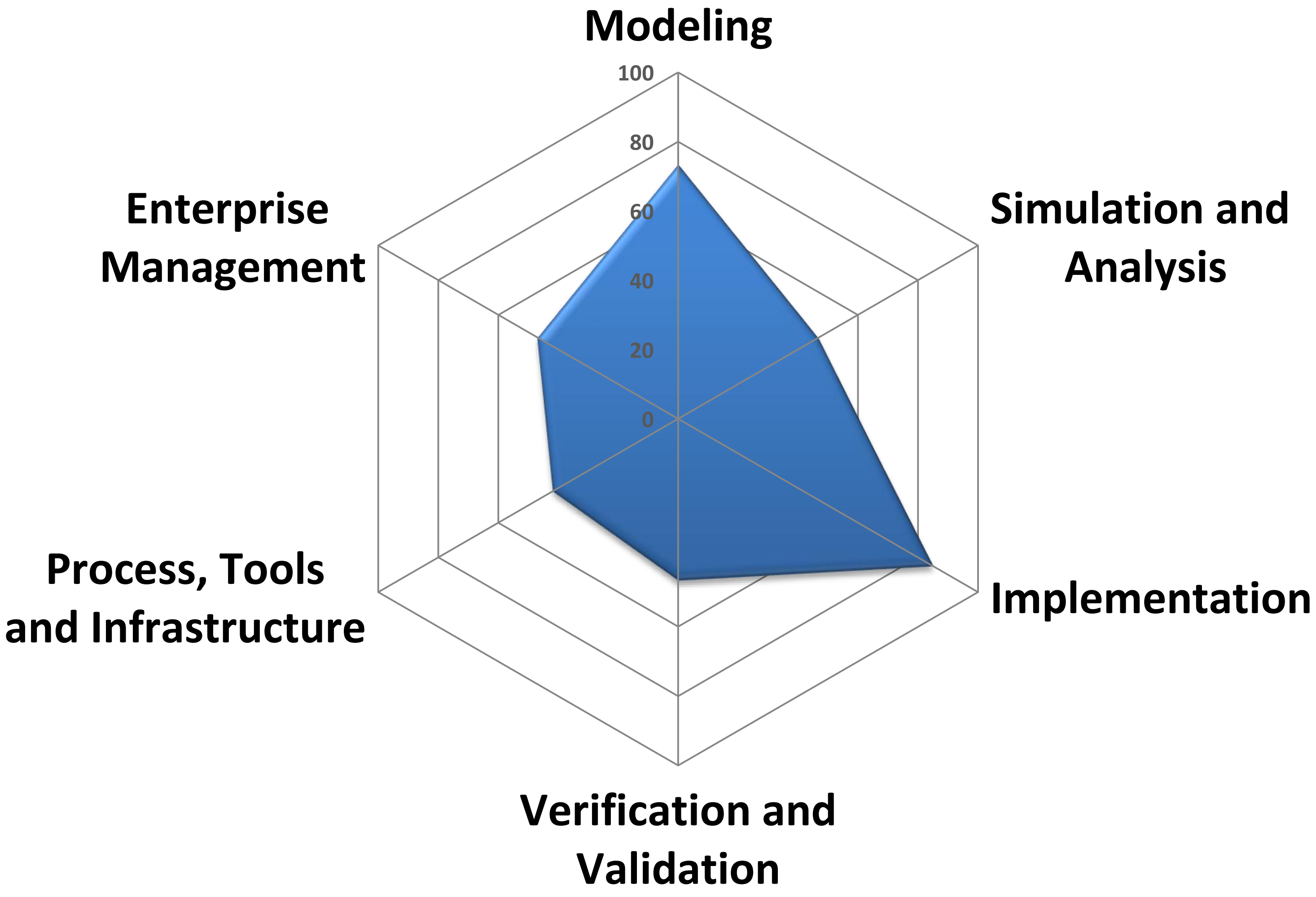
Compilation and Linking

Complete Model Based Design Workflow, Concept to Code

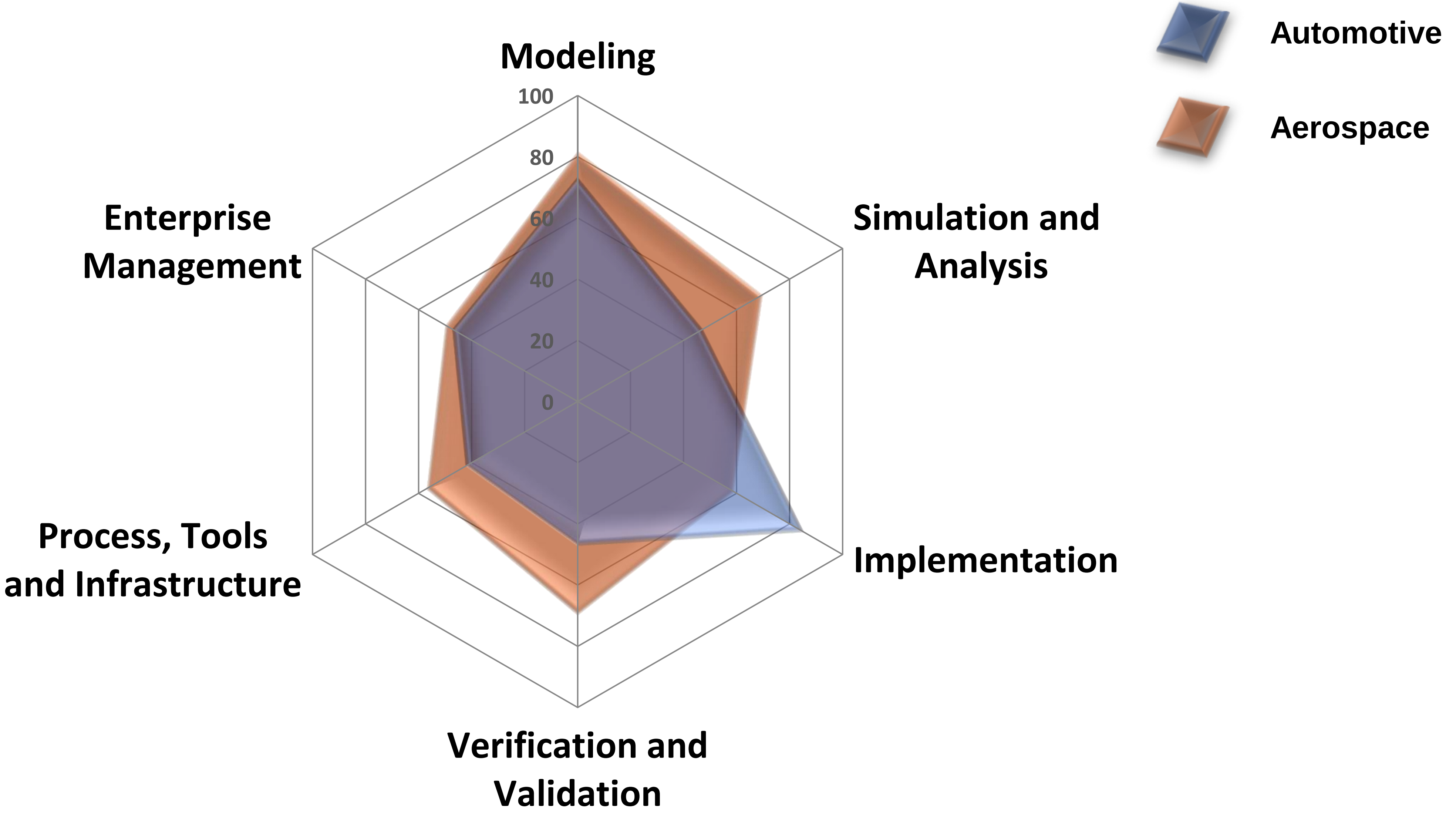
How do you ensure correctness?



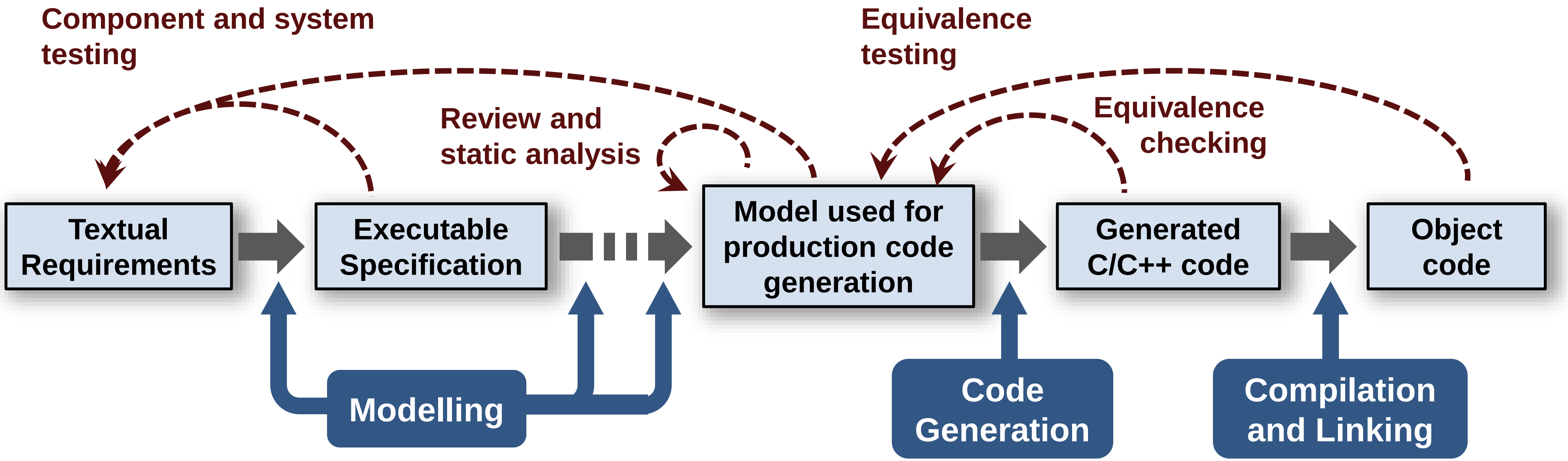
Model-Based Design Maturity, Automotive Industry



Model-Based Design Maturity, Automotive and Aerospace

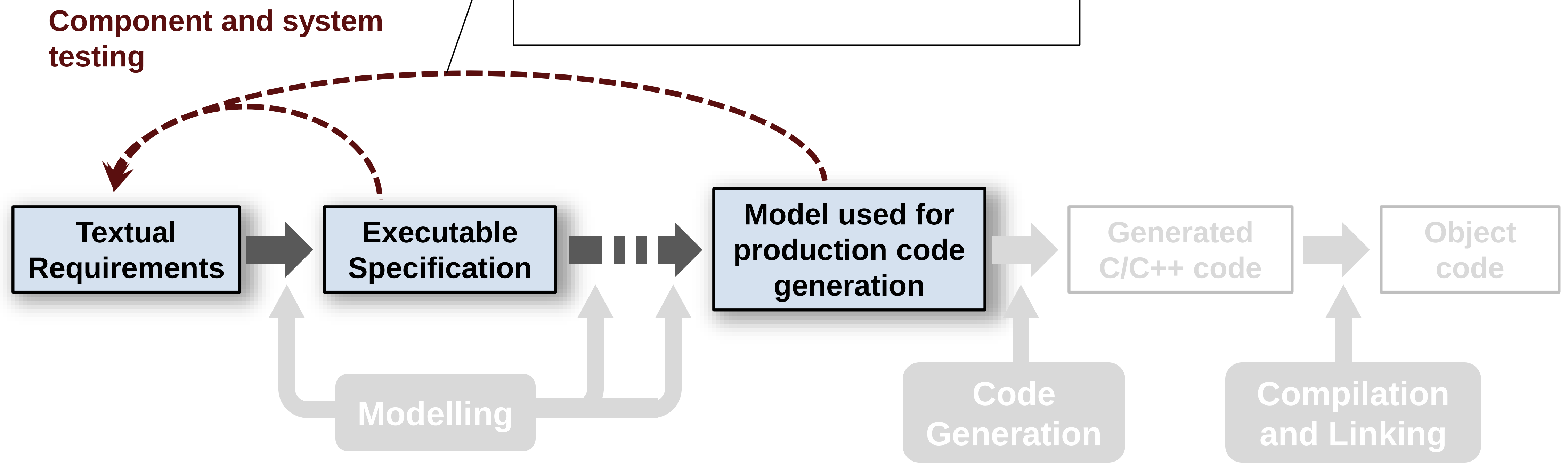


Model Based Design Verification Workflow



Model Based Design Verification Workflow

- Perform simulation
- Link and review requirements
- Isolate and test components
- Measure model coverage
- Address missing coverage
- Property proving



Component and system testing

Textual Requirements

Executable Specification

Model used for production code generation

Generated C/C++ code

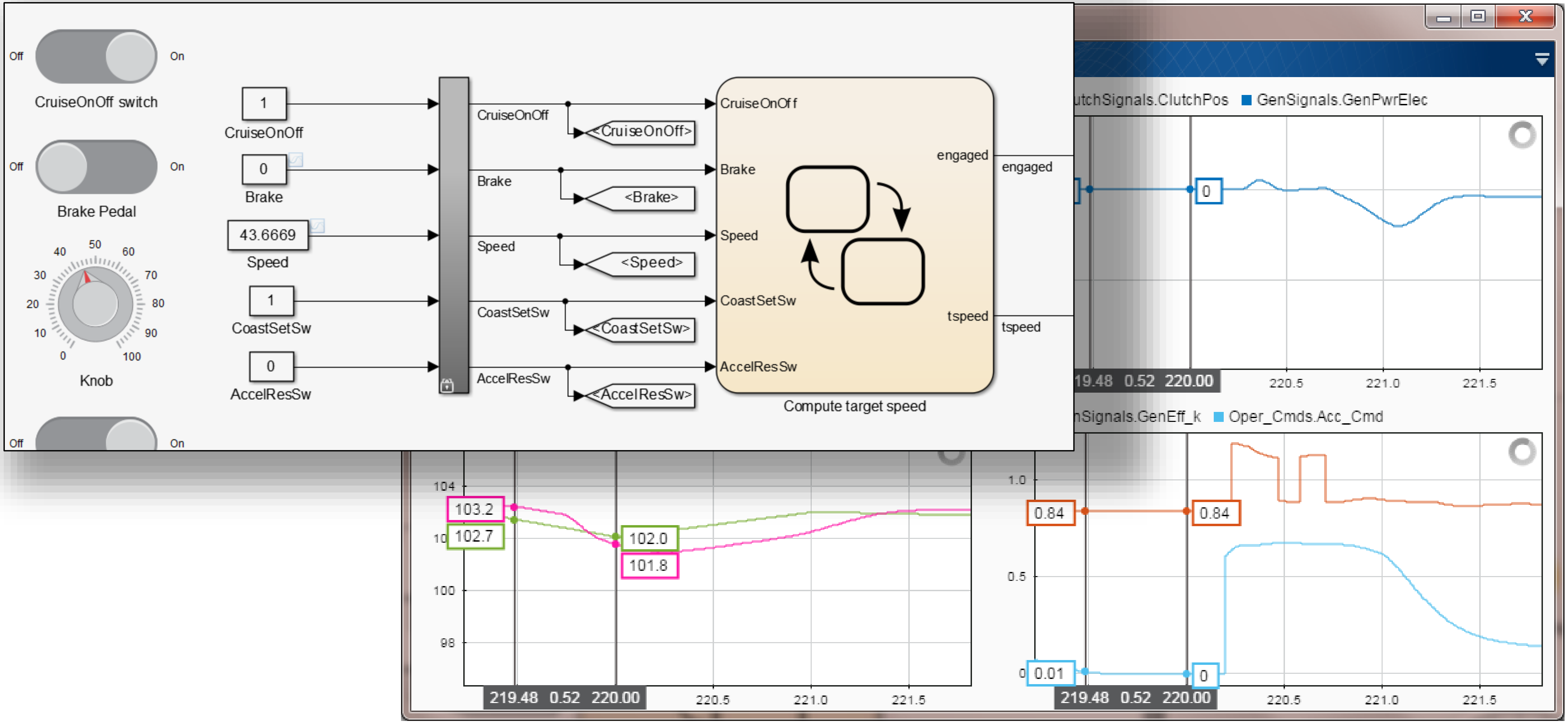
Object code

Modelling

Code Generation

Compilation and Linking

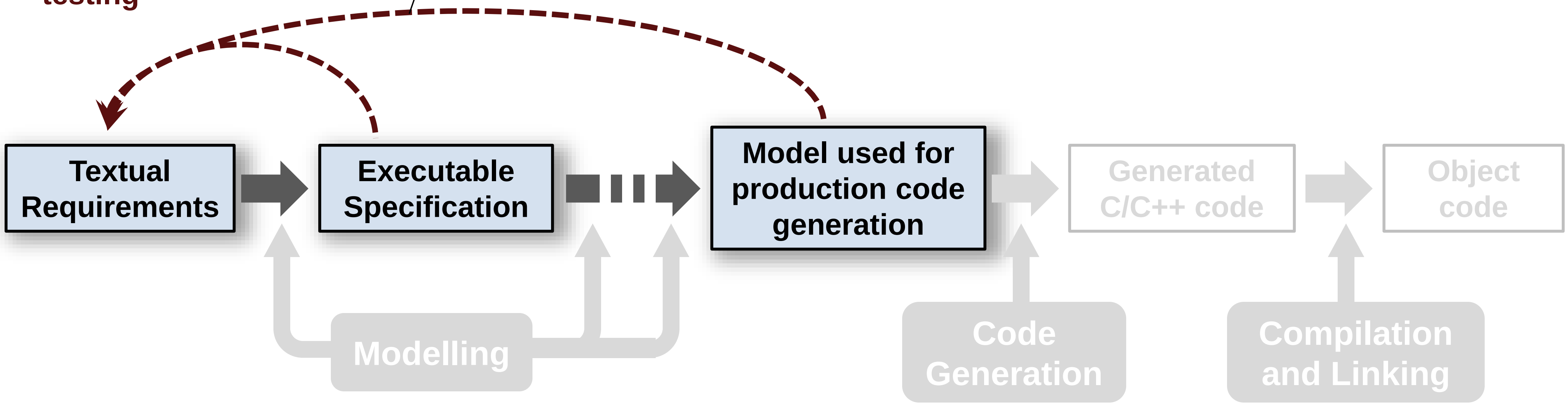
Ad-Hoc Simulation: Explore Behavior Virtually



Model Based Design Verification Workflow

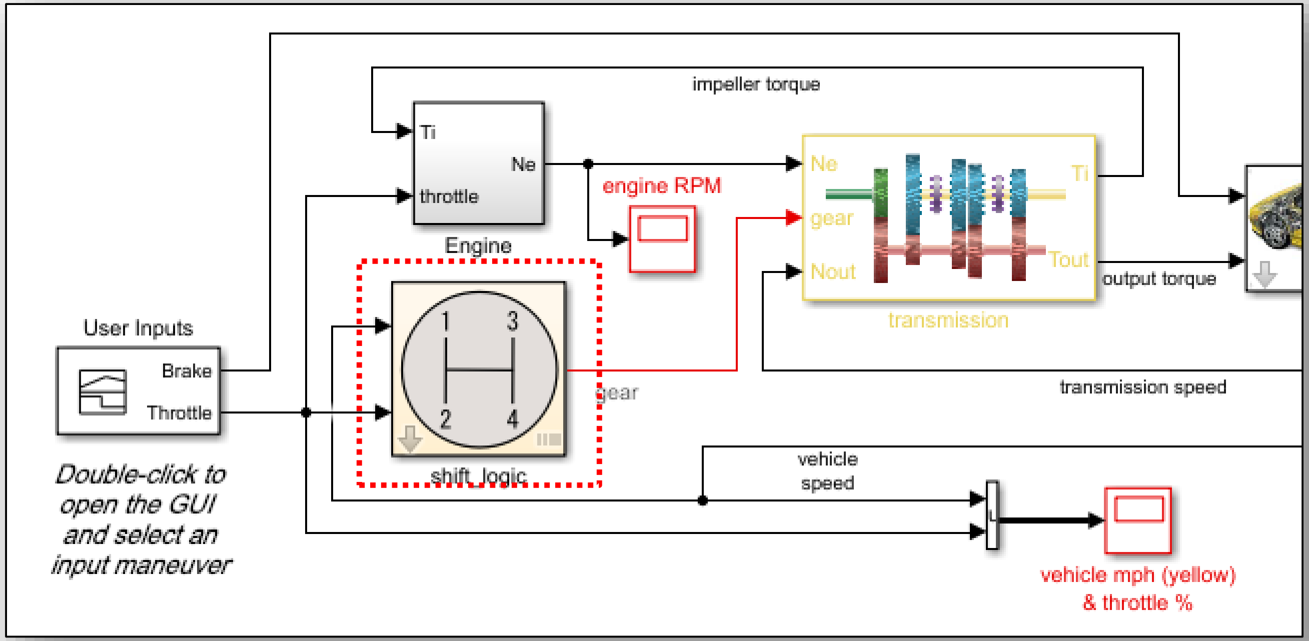
- Perform simulation
- Link and review requirements
- **Isolate and test components**
- **Measure model coverage**
- **Generate tests for missing coverage**
- Manage and organize tests
- Property proving

Component and system testing



Test Harnesses

From any subsystem …



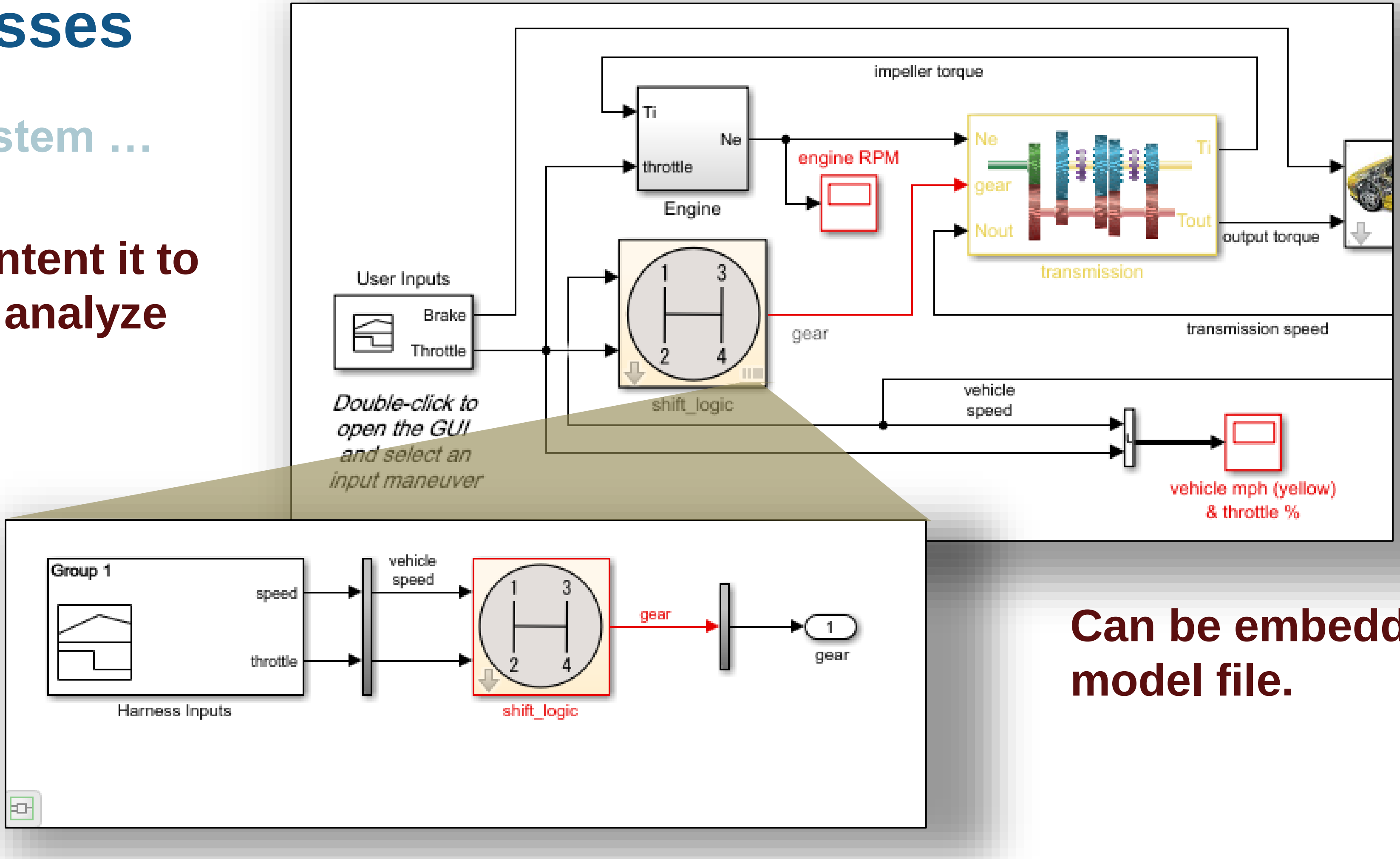
Double-click to open the GUI and select an input maneuver

Test Harnesses

From any subsystem …

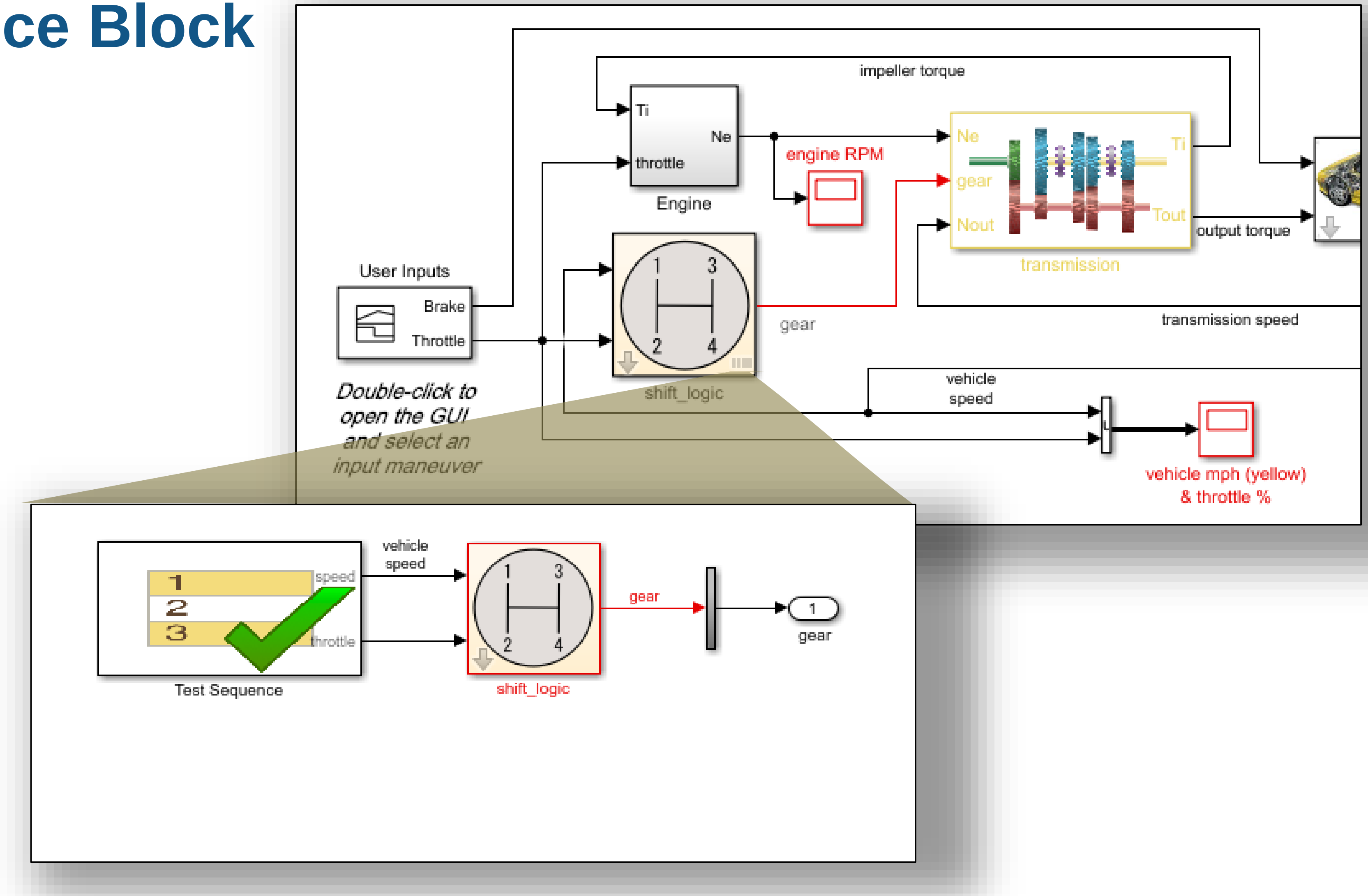
Isolate it with content it to drive inputs and analyze outputs

Simulate independently



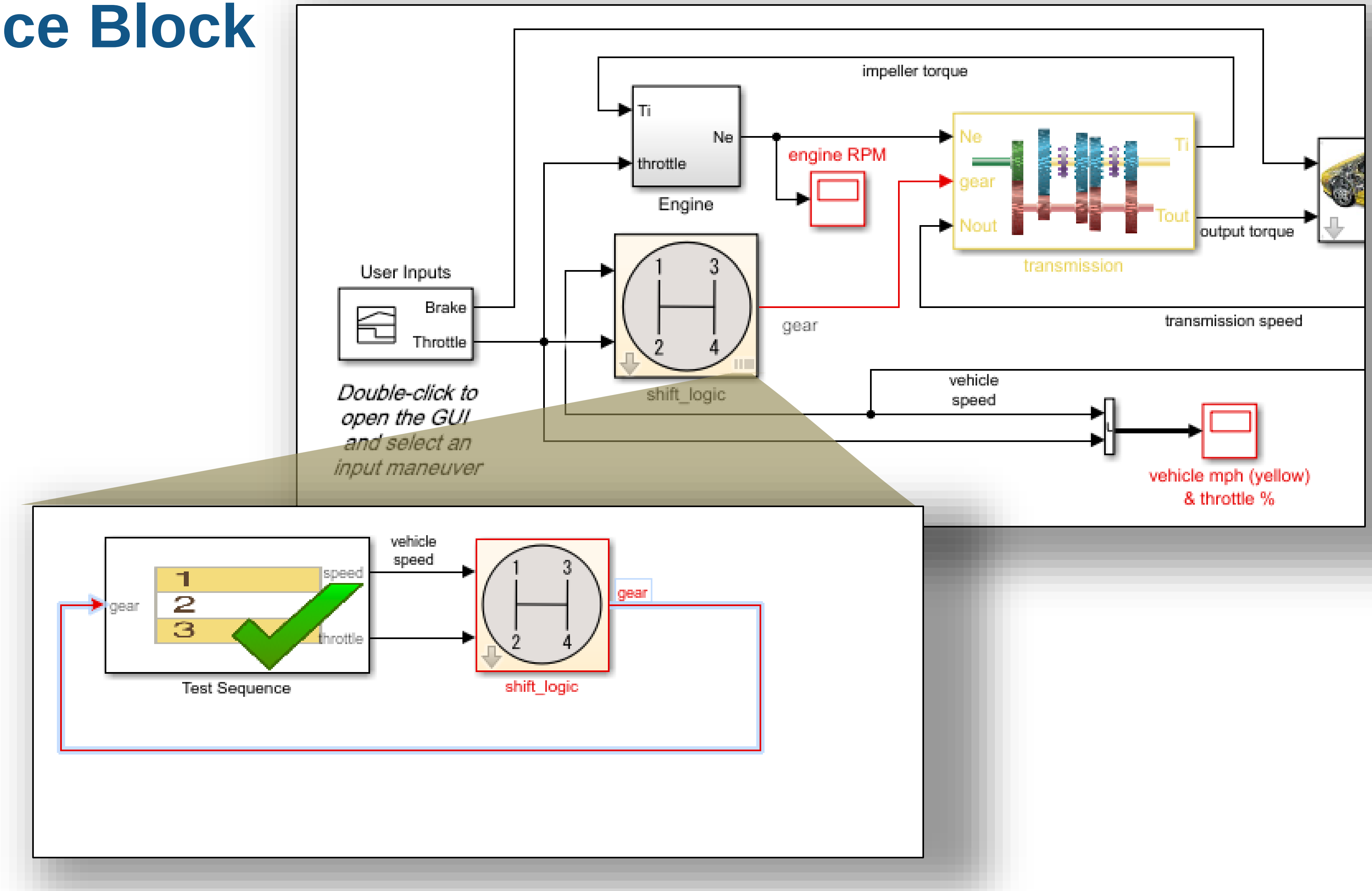
Can be embedded in design model file.

Test Sequence Block



A test sequence block can drive inputs

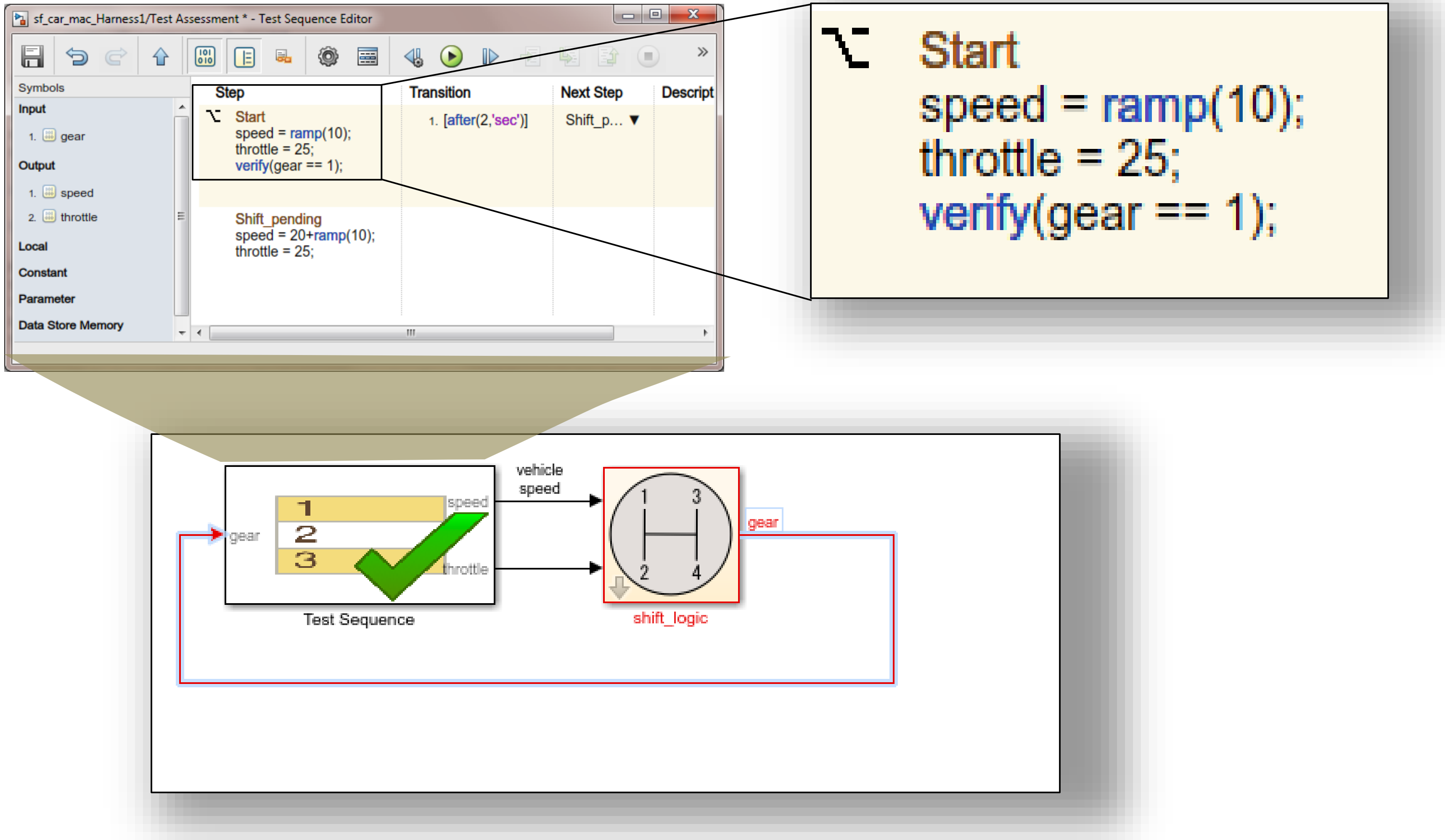
Test Sequence Block



Double-click to open the GUI and select an input maneuver

A test sequence block can drive inputs <u>and asses outputs</u>

Test Sequence Block Syntax



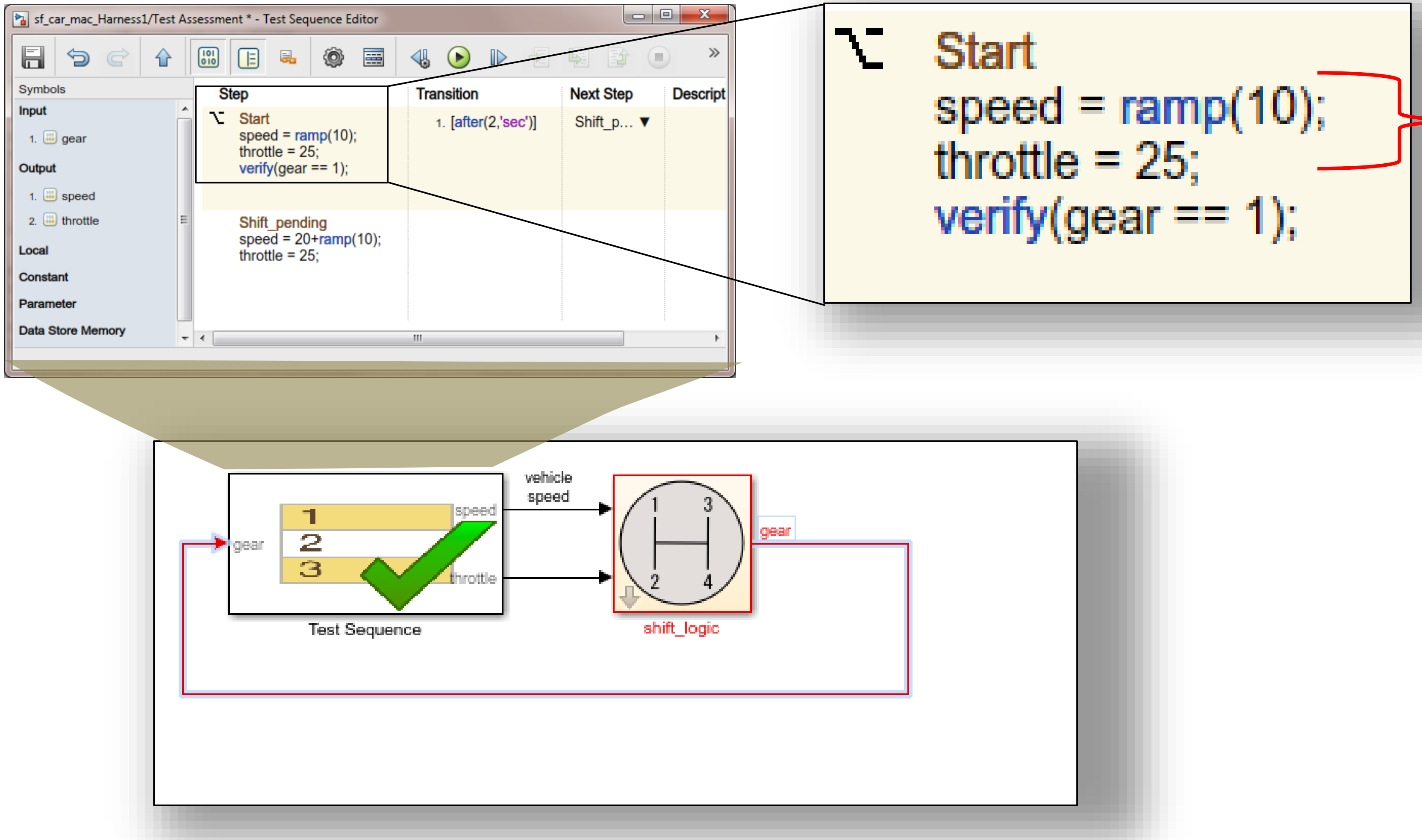
```
Start
speed = ramp(10);
throttle = 25;
verify(gear == 1);
```

Transition: 1. [after(2,'sec')]

Next Step: Shift_p... ▼

```
Shift_pending
speed = 20+ramp(10);
throttle = 25;
```

Test Sequence Block Syntax



sf_car_mac_Harness1/Test Assessment * - Test Sequence Editor

Symbols

- Input
  1. gear
- Output
  1. speed
  2. throttle
- Local
- Constant
- Parameter
- Data Store Memory

Step	Transition	Next Step	Descript

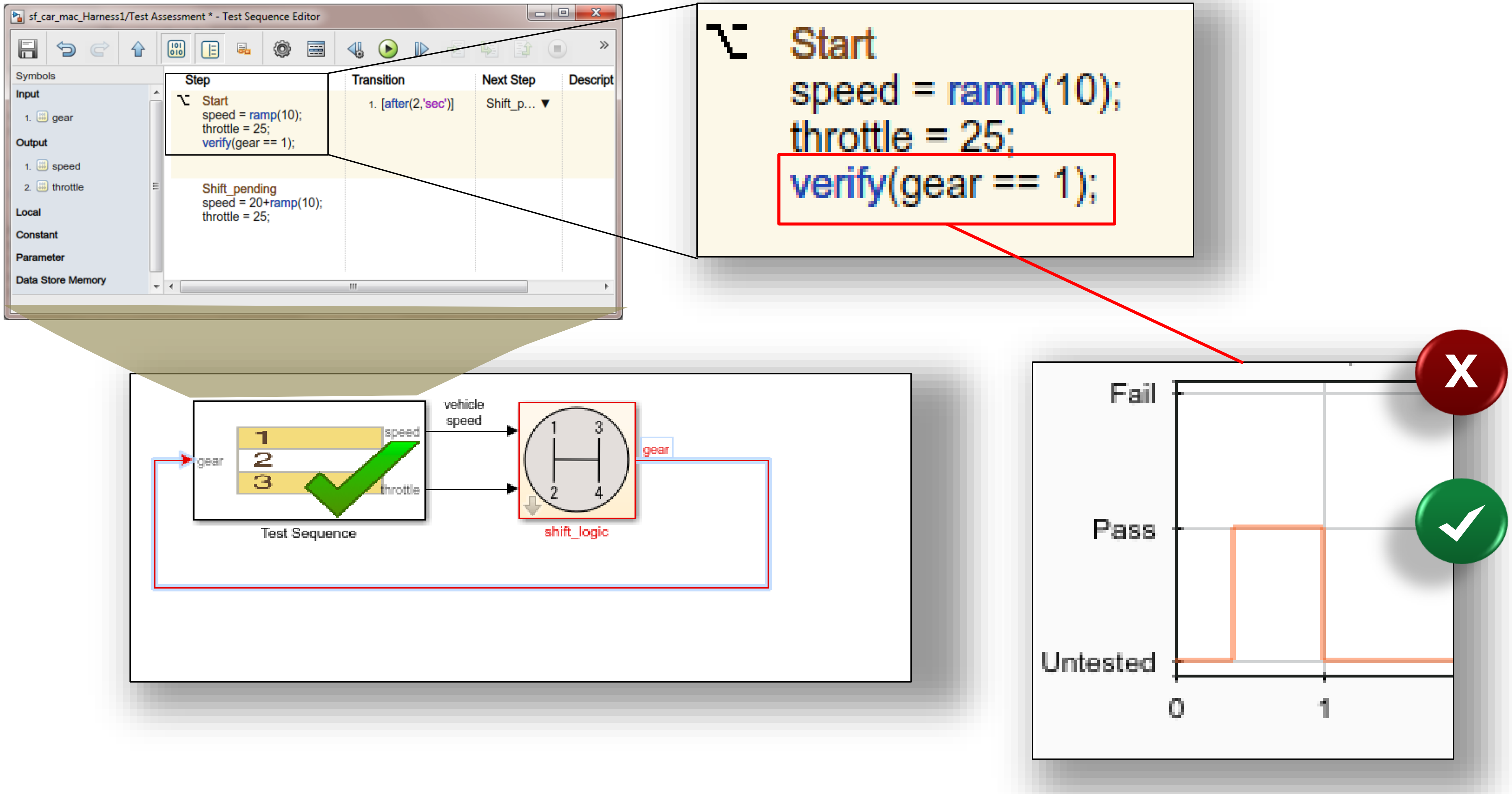
Start
speed = ramp(10);
throttle = 25;
verify(gear == 1);
	1. [after(2,'sec')]	Shift_p... ▼	

Shift_pending
speed = 20+ramp(10);
throttle = 25;			

```
Start
speed = ramp(10);
throttle = 25;
verify(gear == 1);
```

Define Inputs

Defining Pass/Fail Criteria



Step	Transition	Next Step	Descript

Start
speed = ramp(10);
throttle = 25;
verify(gear == 1);
	1. [after(2,'sec')]	Shift_p... ▼	

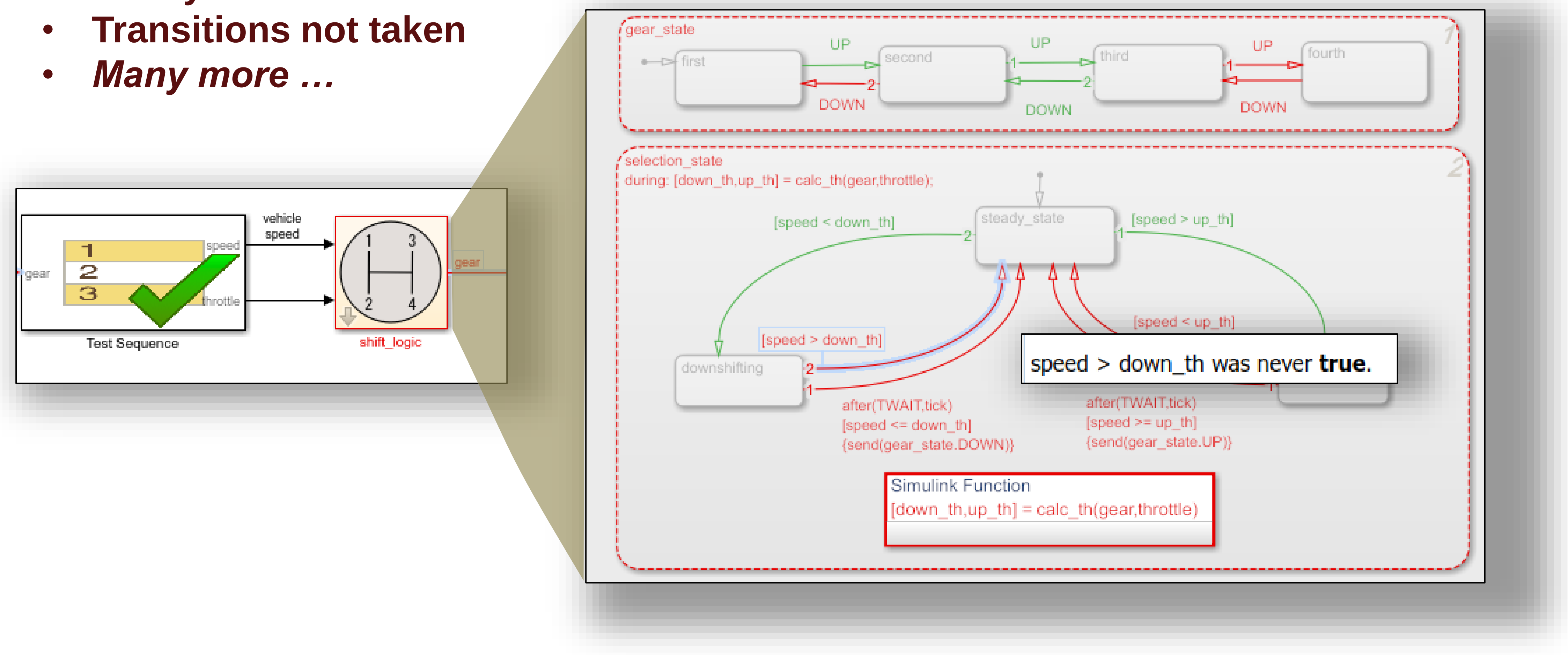
Shift_pending
speed = 20+ramp(10);
throttle = 25;			

```
Start
speed = ramp(10);
throttle = 25;
verify(gear == 1);
```

Model Coverage

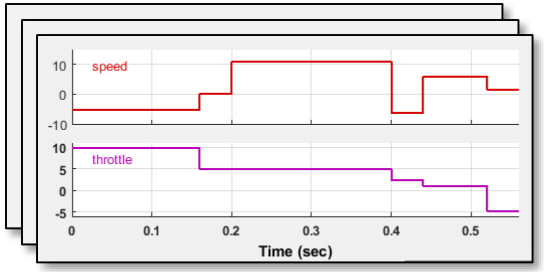
Identify testing gaps:

- **Untested switch positions**
- **Subsystems not executed**
- **Transitions not taken**
- ***Many more …***

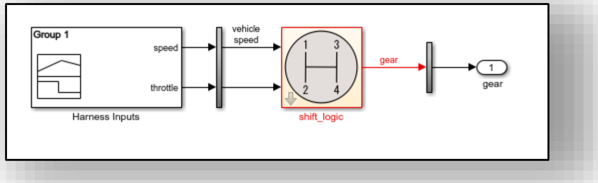


speed > down_th was never **true.**

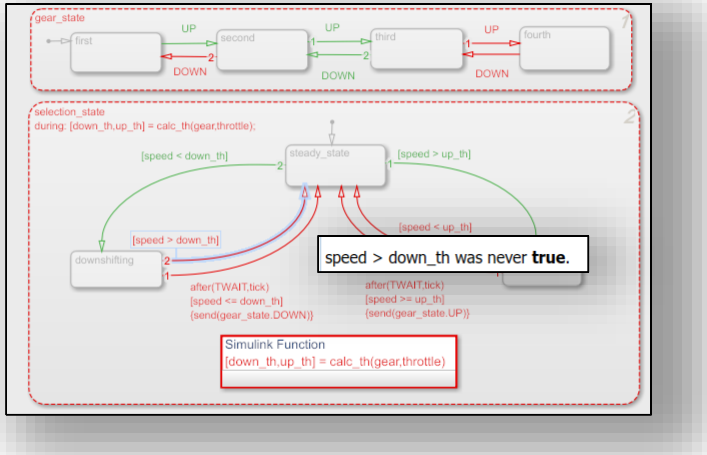
Addressing Missing Coverage



Test Cases

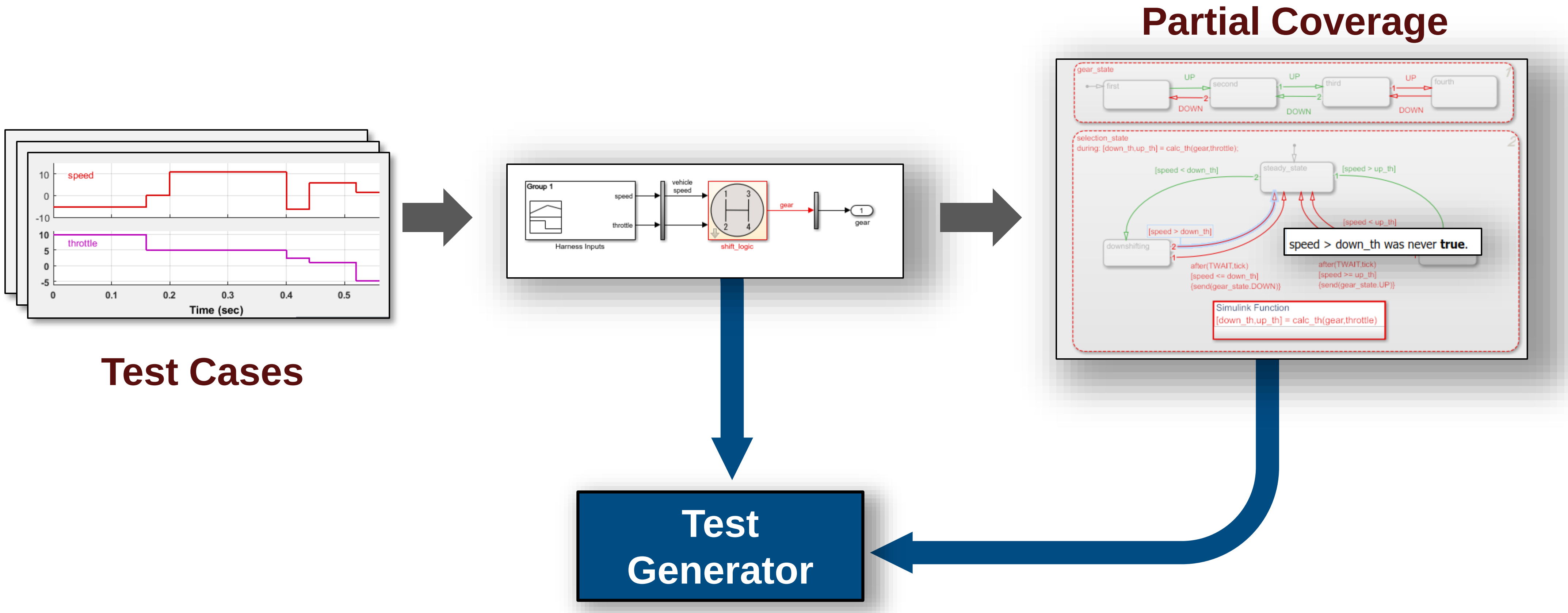


Partial Coverage



speed > down_th was never **true**.

Addressing Missing Coverage



Partial Coverage

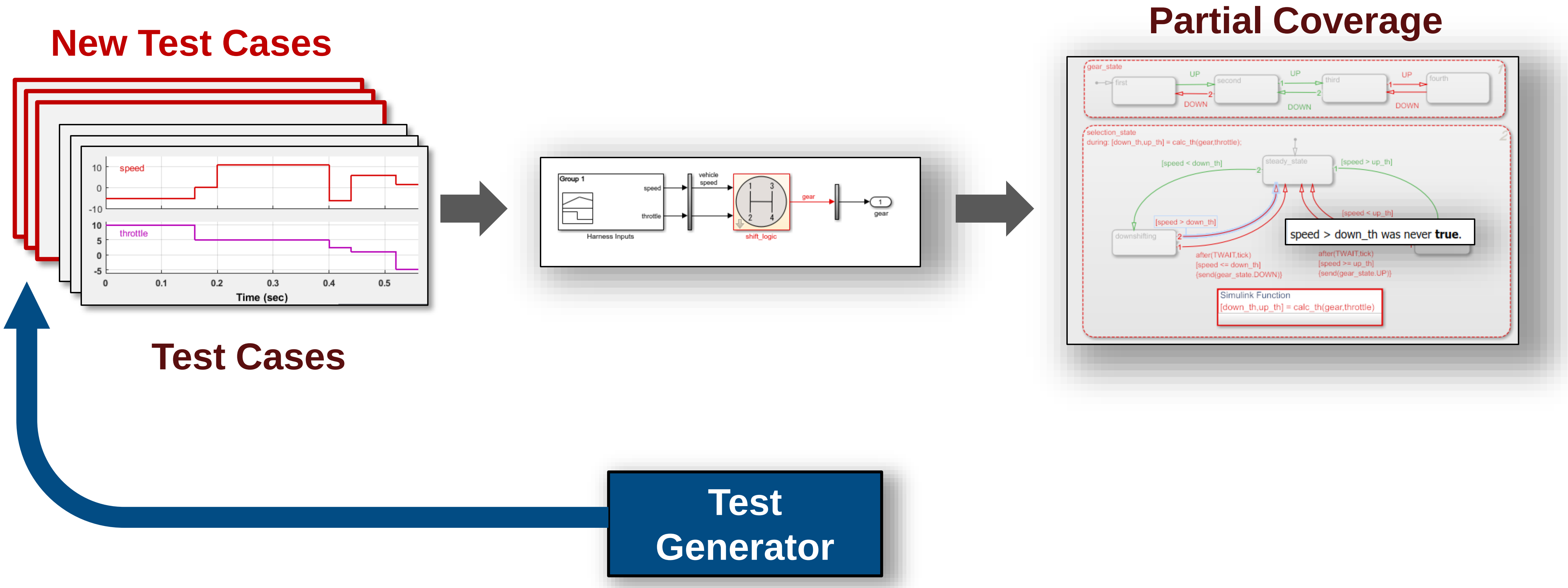
speed > down_th was never true.

Test Cases

Test Generator

Simulink Design Verifier

Addressing Missing Coverage



New Test Cases

Test Cases

Partial Coverage

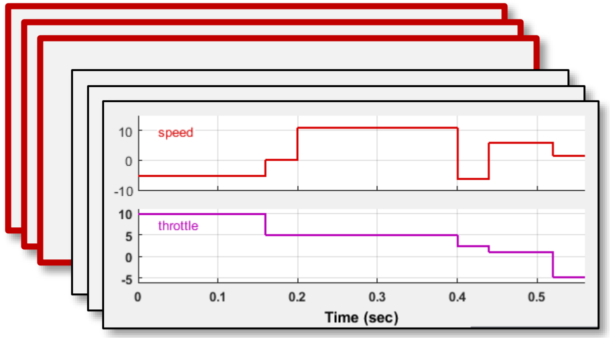
speed > down_th was never **true**.

Test Generator

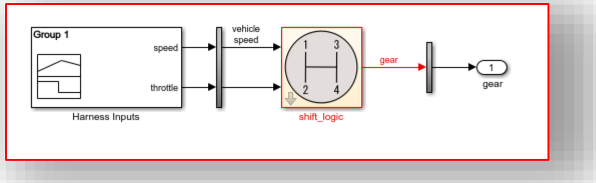
Simulink Design Verifier

Addressing Missing Coverage

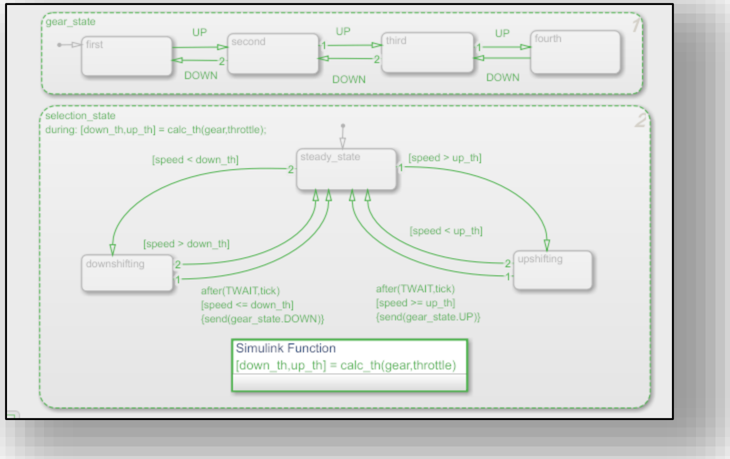
New Test Cases



Test Cases

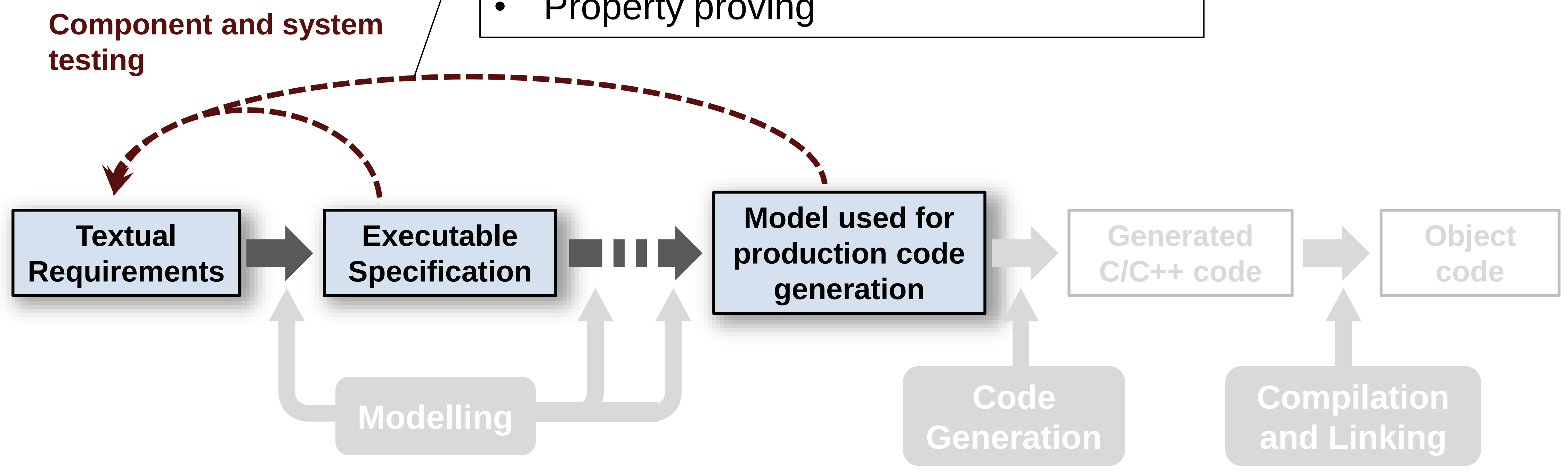


Full Coverage



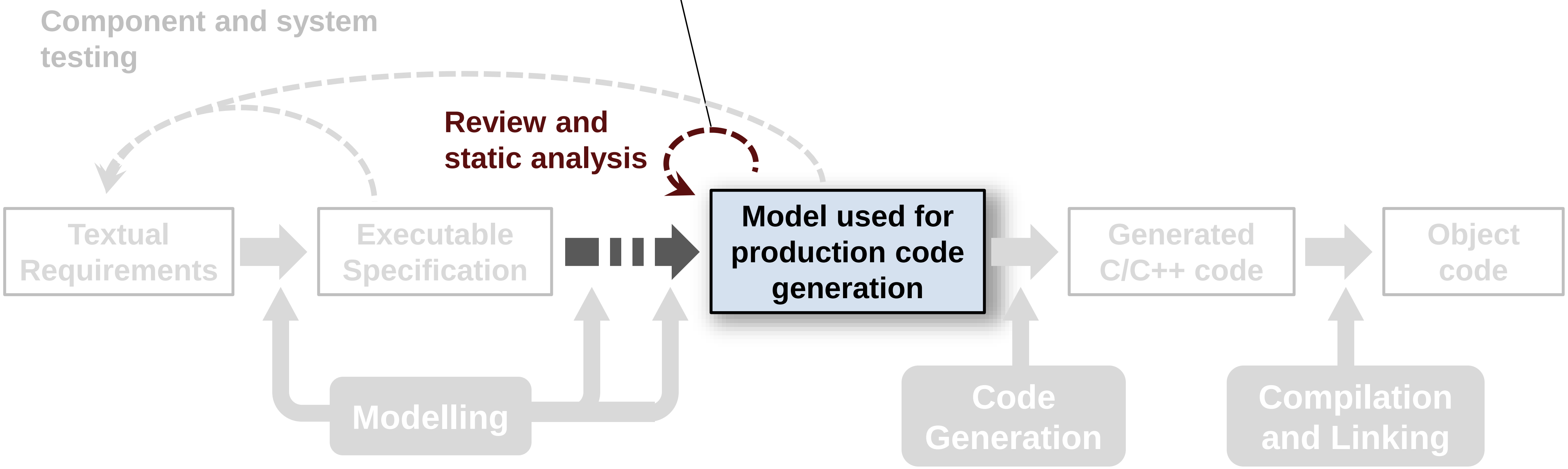
Model Based Design Verification Workflow

- Perform simulation
- Link and review requirements
- **Isolate and test components**
- **Measure model coverage**
- **Generate tests for missing coverage**
- Manage and organize tests
- Property proving

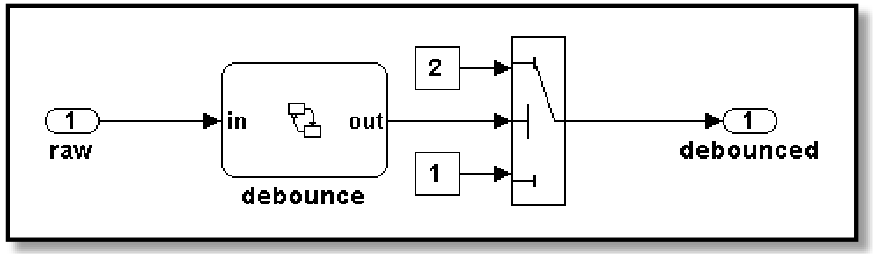


Model Based Design Verification Workflow

- Manual review
- Standards compliance checking
- **Design error detection**
- Complexity analysis

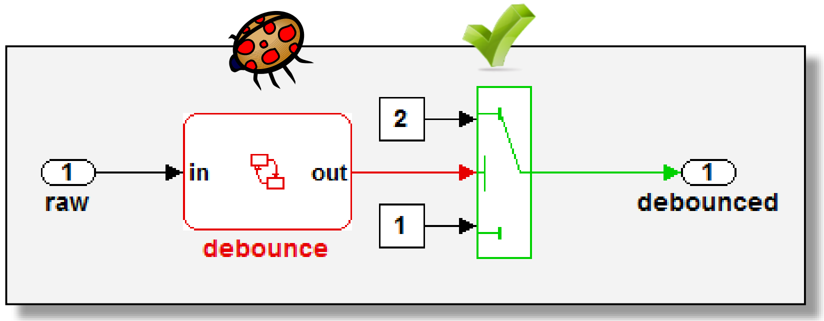


Detecting Hidden Run-Time Design Errors



Design Model

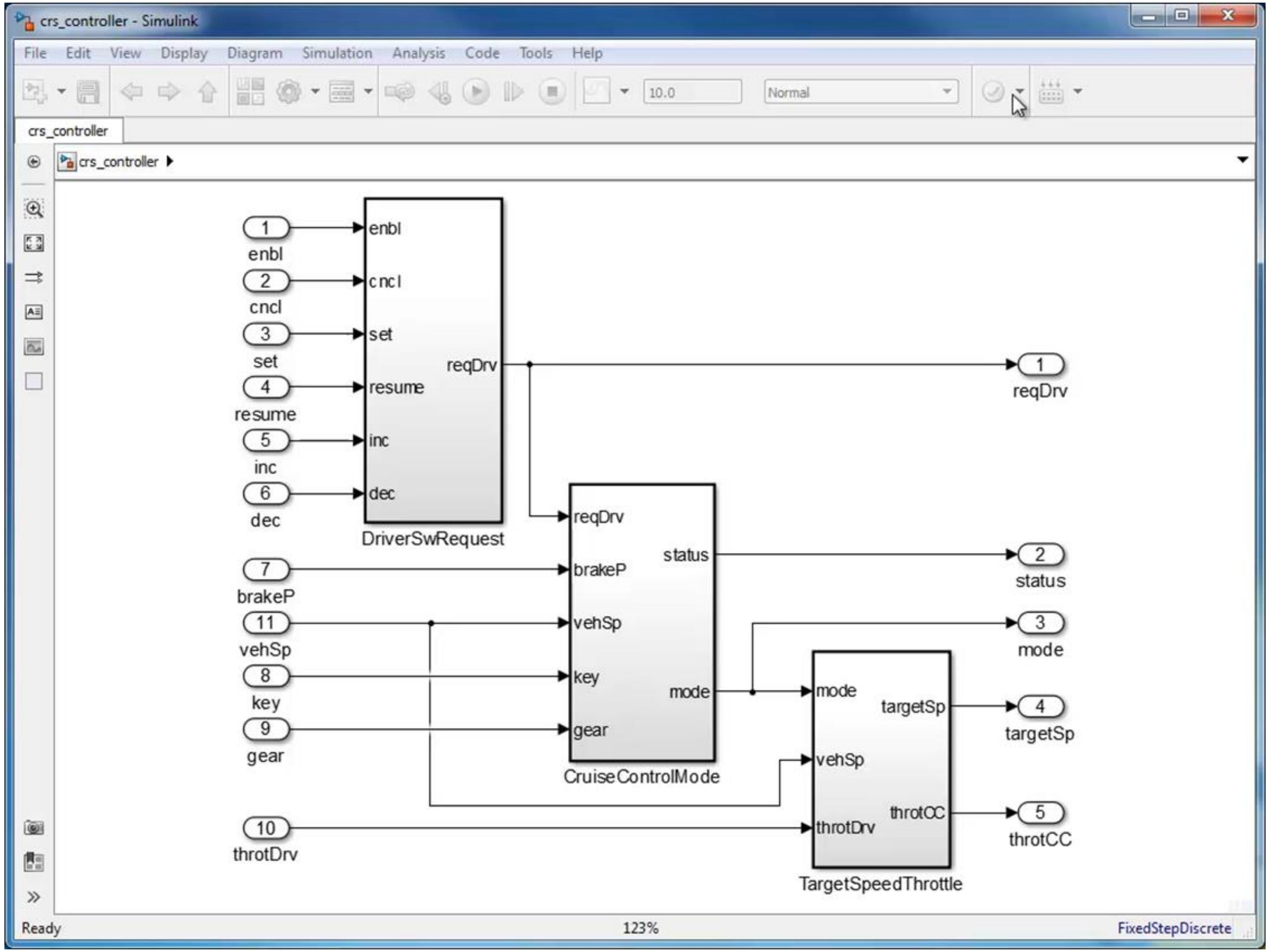
Design error detection



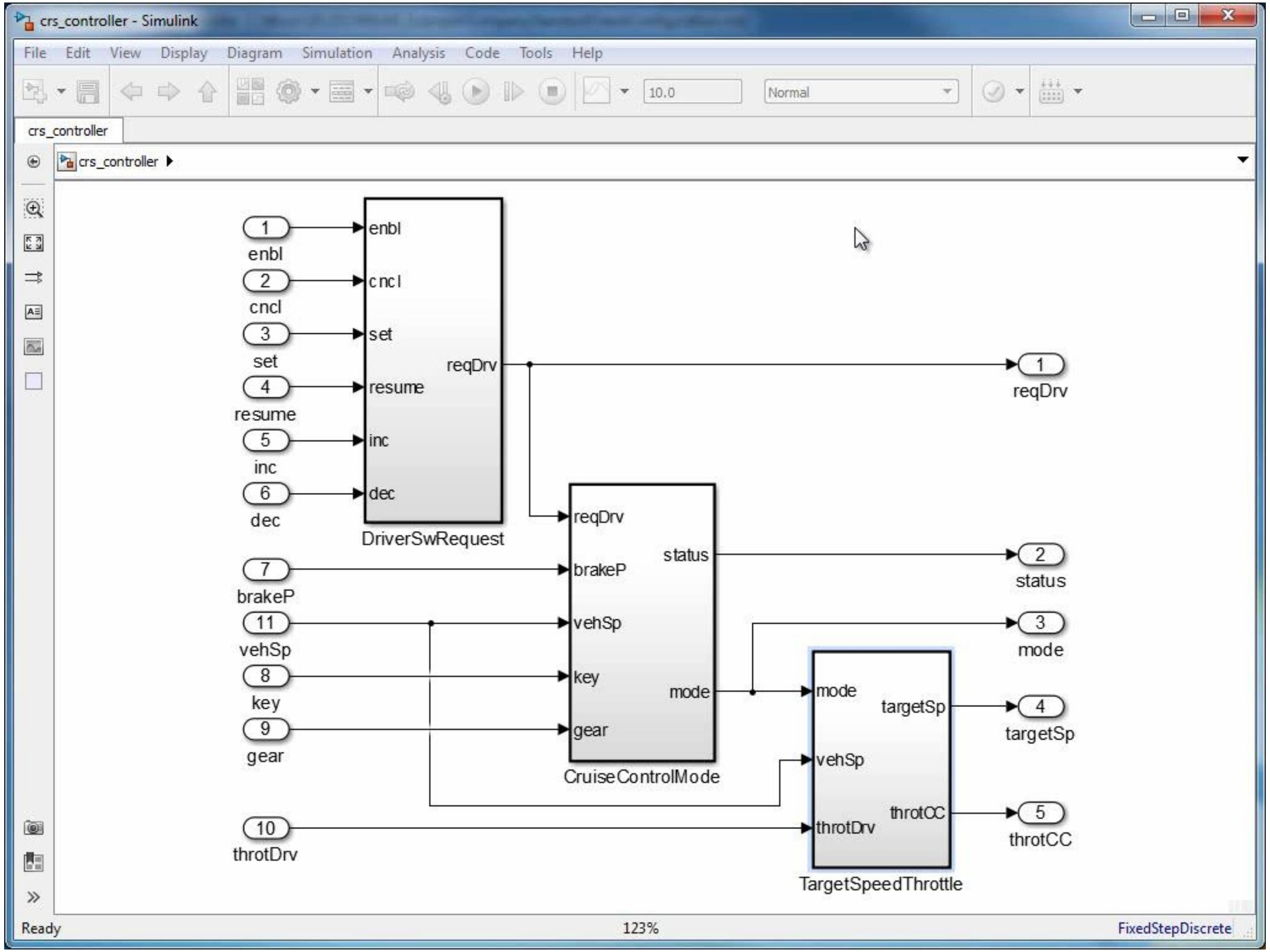
Highlighted Model

- Integer overflow
- Division by zero
- Array out-of-bounds
- Range violations
- Dead Logic

Detecting Hidden Run-Time Design Errors

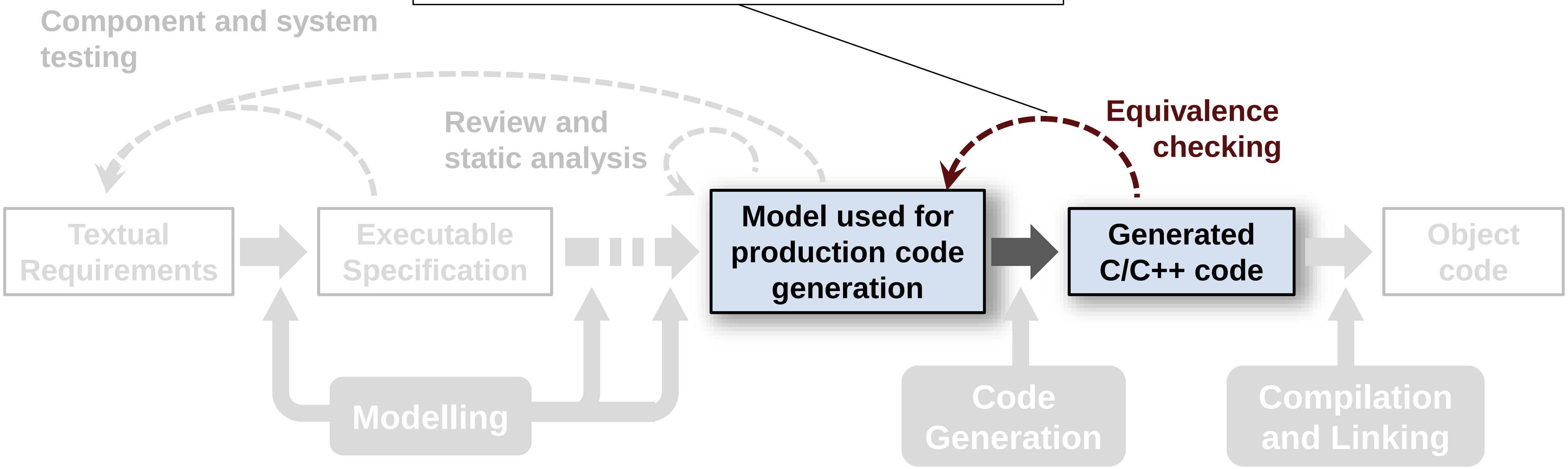


Detecting Hidden Run-Time Design Errors

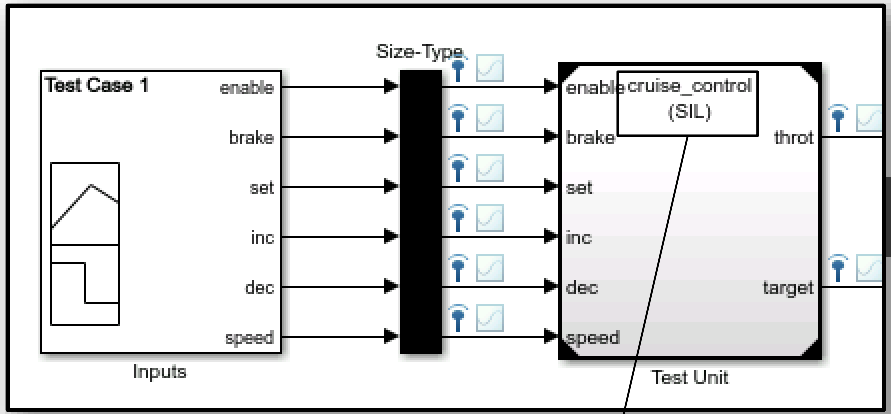


Model Based Design Verification Workflow

- Perform SIL Testing
- **Measure code coverage**
- Verify code with Polyspace
- Verify consistency with Simulink Code Inspector

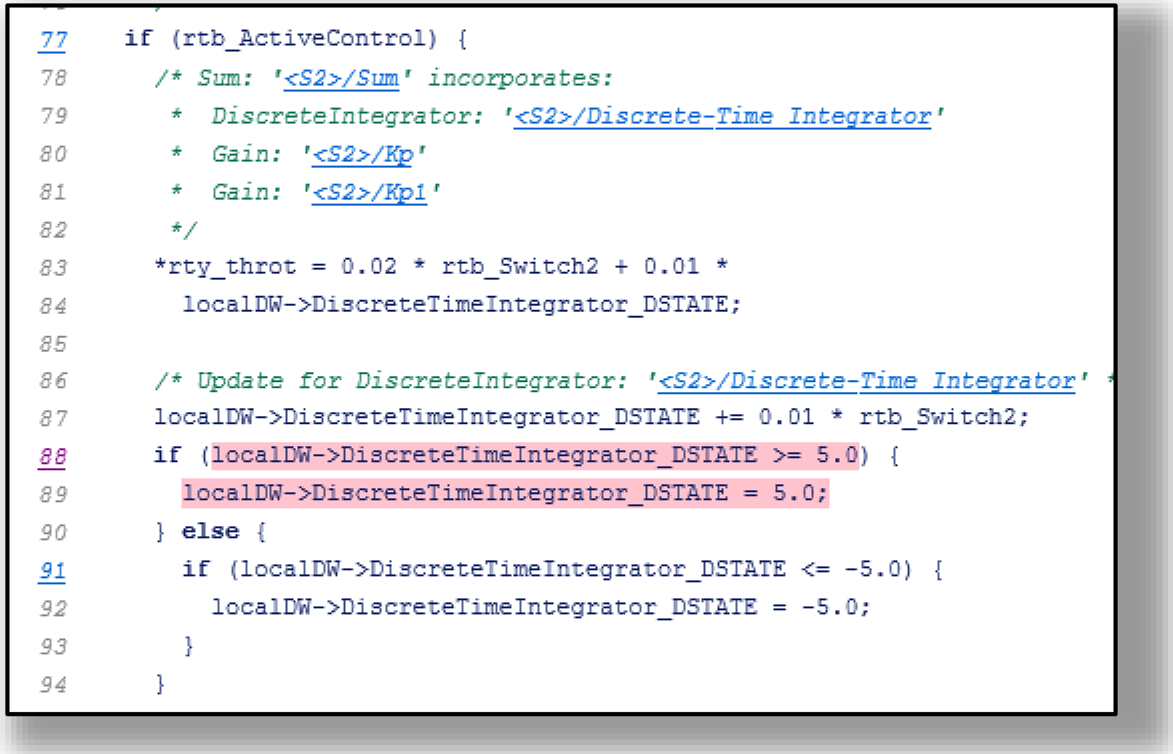


Coverage for Generated Code (R2016a)



cruise_control (SIL)

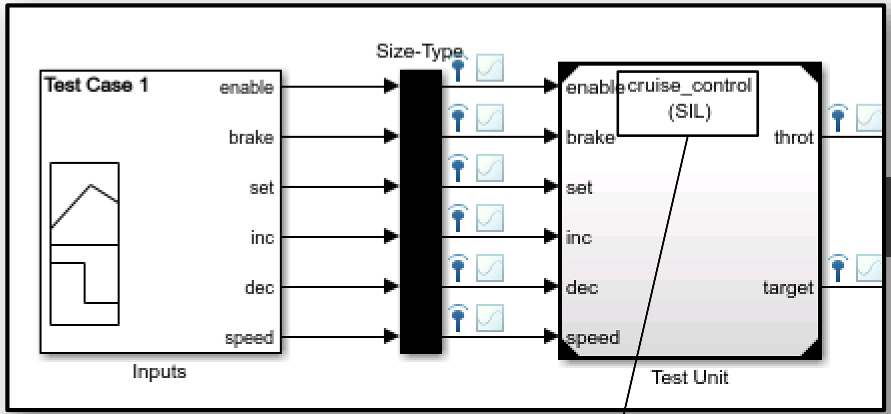
Press Play



```
if (rtb_ActiveControl) {
  /* Sum: '<S2>/Sum' incorporates:
   *  DiscreteIntegrator: '<S2>/Discrete-Time Integrator'
   *  Gain: '<S2>/Kp'
   *  Gain: '<S2>/Kp1'
   */
  *rty_throt = 0.02 * rtb_Switch2 + 0.01 *
    localDW->DiscreteTimeIntegrator_DSTATE;

  /* Update for DiscreteIntegrator: '<S2>/Discrete-Time Integrator' *
  localDW->DiscreteTimeIntegrator_DSTATE += 0.01 * rtb_Switch2;
  if (localDW->DiscreteTimeIntegrator_DSTATE >= 5.0) {
    localDW->DiscreteTimeIntegrator_DSTATE = 5.0;
  } else {
    if (localDW->DiscreteTimeIntegrator_DSTATE <= -5.0) {
      localDW->DiscreteTimeIntegrator_DSTATE = -5.0;
    }
  }
```

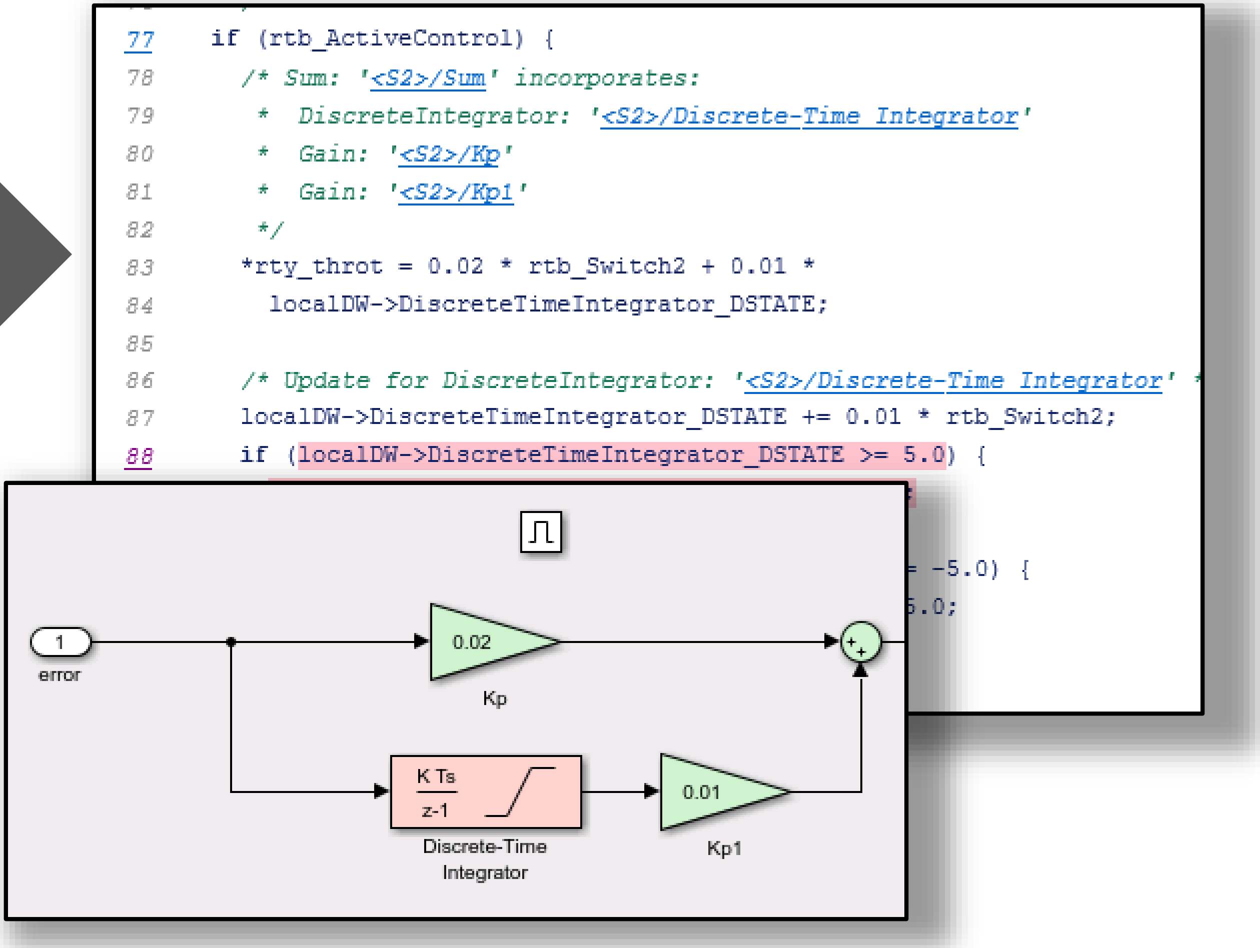
Generated Code Coverage

Coverage for Generated Code (R2016a)



cruise_control (SIL)

Press Play

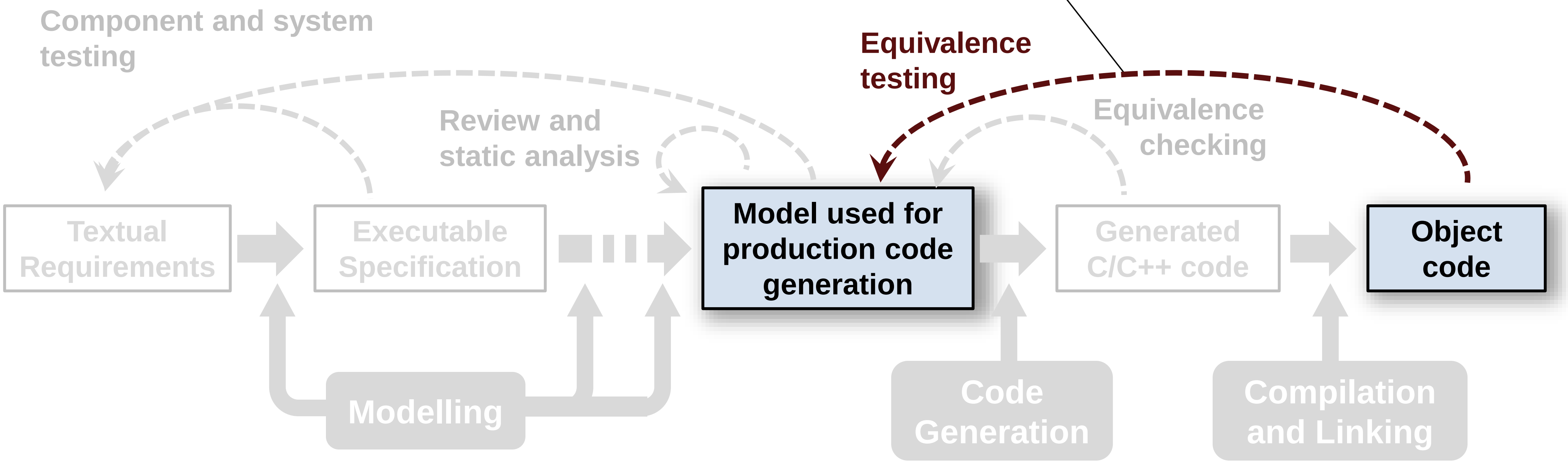


```
    if (rtb_ActiveControl) {
      /* Sum: '<S2>/Sum' incorporates:
       *  DiscreteIntegrator: '<S2>/Discrete-Time Integrator'
       *  Gain: '<S2>/Kp'
       *  Gain: '<S2>/Kp1'
       */
      *rty_throt = 0.02 * rtb_Switch2 + 0.01 *
        localDW->DiscreteTimeIntegrator_DSTATE;

      /* Update for DiscreteIntegrator: '<S2>/Discrete-Time Integrator' *
      localDW->DiscreteTimeIntegrator_DSTATE += 0.01 * rtb_Switch2;
      if (localDW->DiscreteTimeIntegrator_DSTATE >= 5.0) {

                                                                = -5.0) {
                                                               5.0;
```

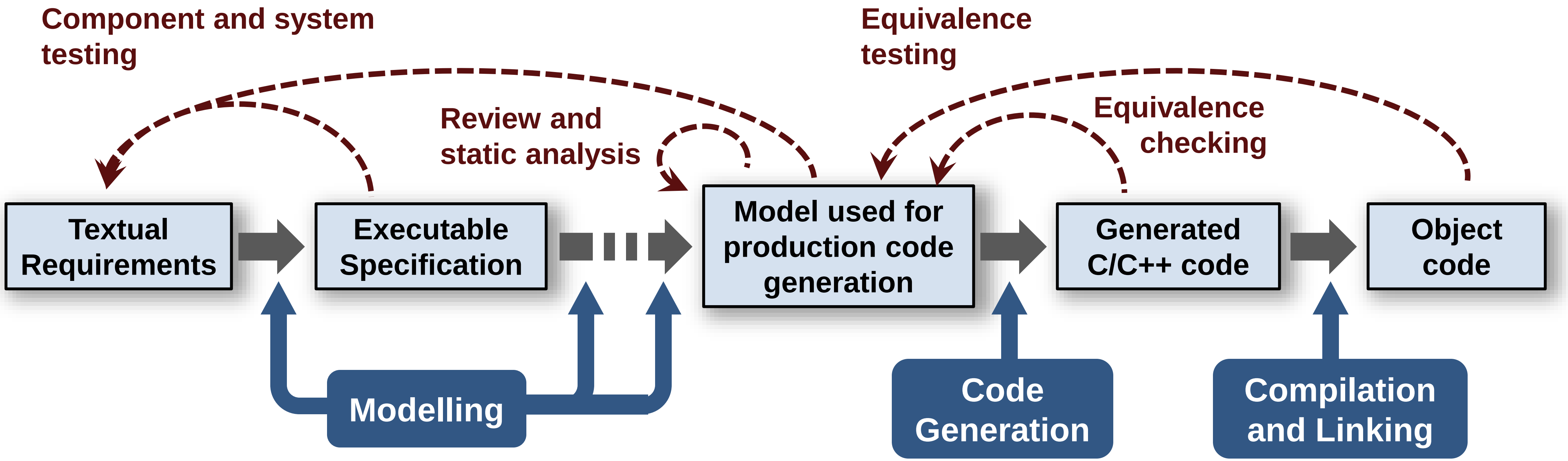
Can also be highlighted on model

Model Based Design Verification Workflow

- Perform PIL Testing
- Perform HIL Testing

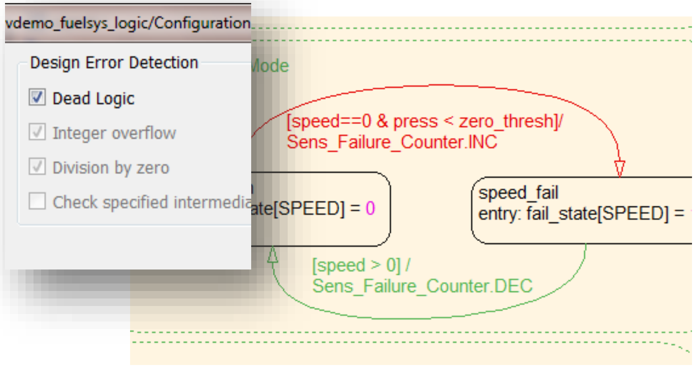


Model Based Design Verification Workflow



Systematic Verification

- Ensure that verification is systematically performed across:
  - All requirements
  - Complete model structure
  - Complete code structure
  - All design behaviors


vdemo_fuelsys_logic/Configuration

Design Error Detection

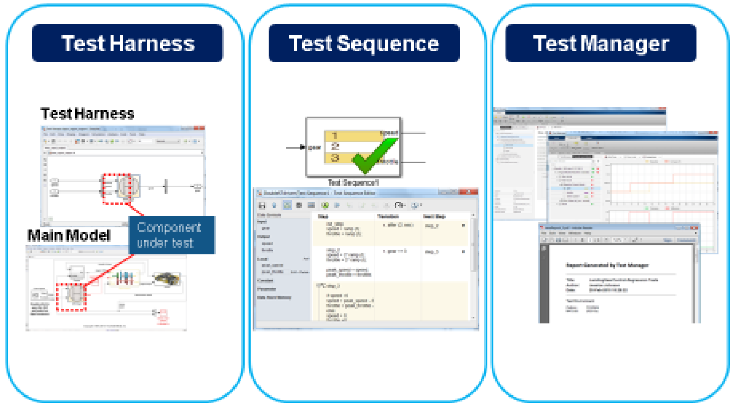
☑ Dead Logic

☑ Integer overflow

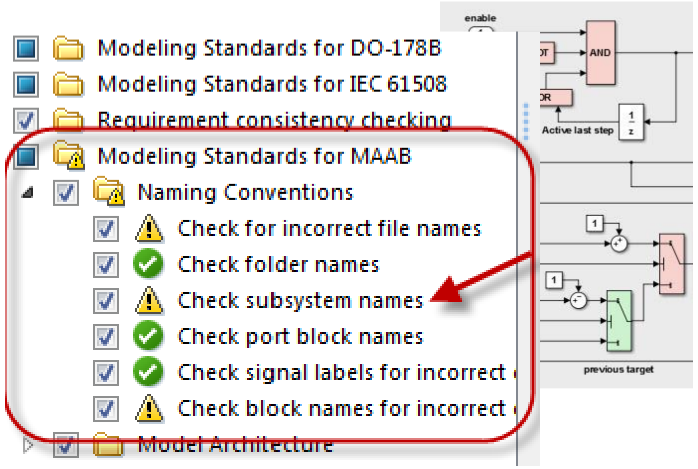
☑ Division by zero

☐ Check specified intermedi

Simulink Design Verifier



Simulink Test

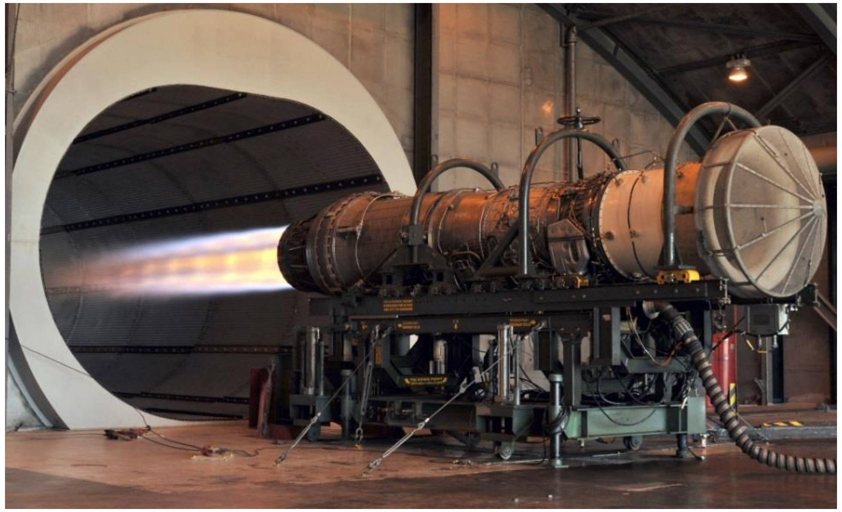

- Modeling Standards for DO-178B
- Modeling Standards for IEC 61508
- ☑ Requirement consistency checking
- Modeling Standards for MAAB
  - ☑ Naming Conventions
    - ☑ ⚠ Check for incorrect file names
    - ☑ ✓ Check folder names
    - ☑ ⚠ Check subsystem names
    - ☑ ✓ Check port block names
    - ☑ ✓ Check signal labels for incorrect
    - ☑ ⚠ Check block names for incorrect
  - ☑ Model Architecture

Simulink Verification & Validation

Test and Verification

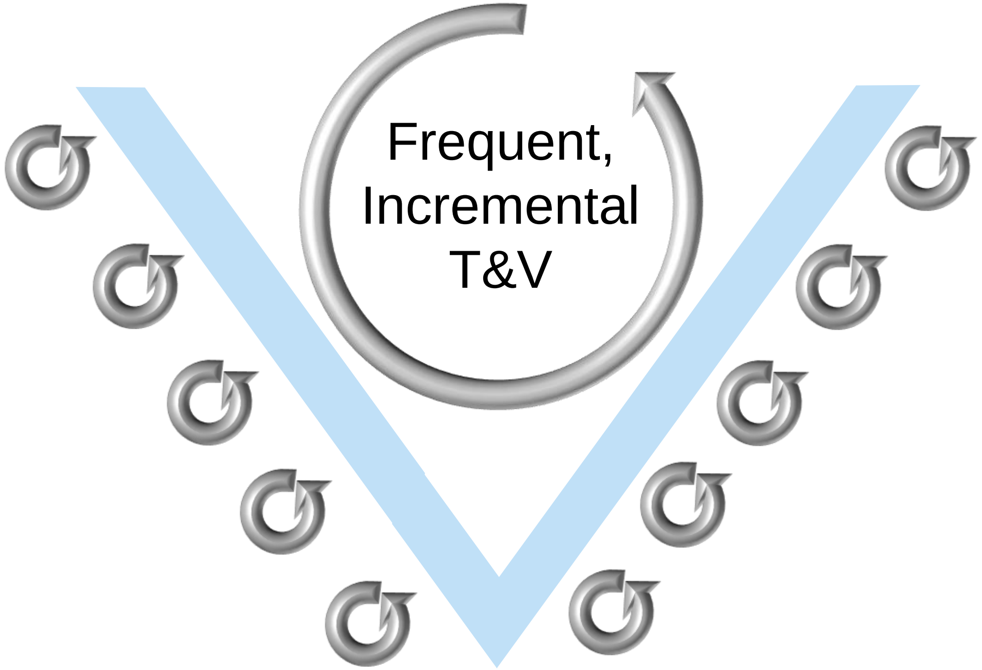
- Essential
- Expensive
- Complex

Pain Points



Test and Verification

- Essential → More Complete
- Expensive → Faster
- Complex → Simpler



Thank You!