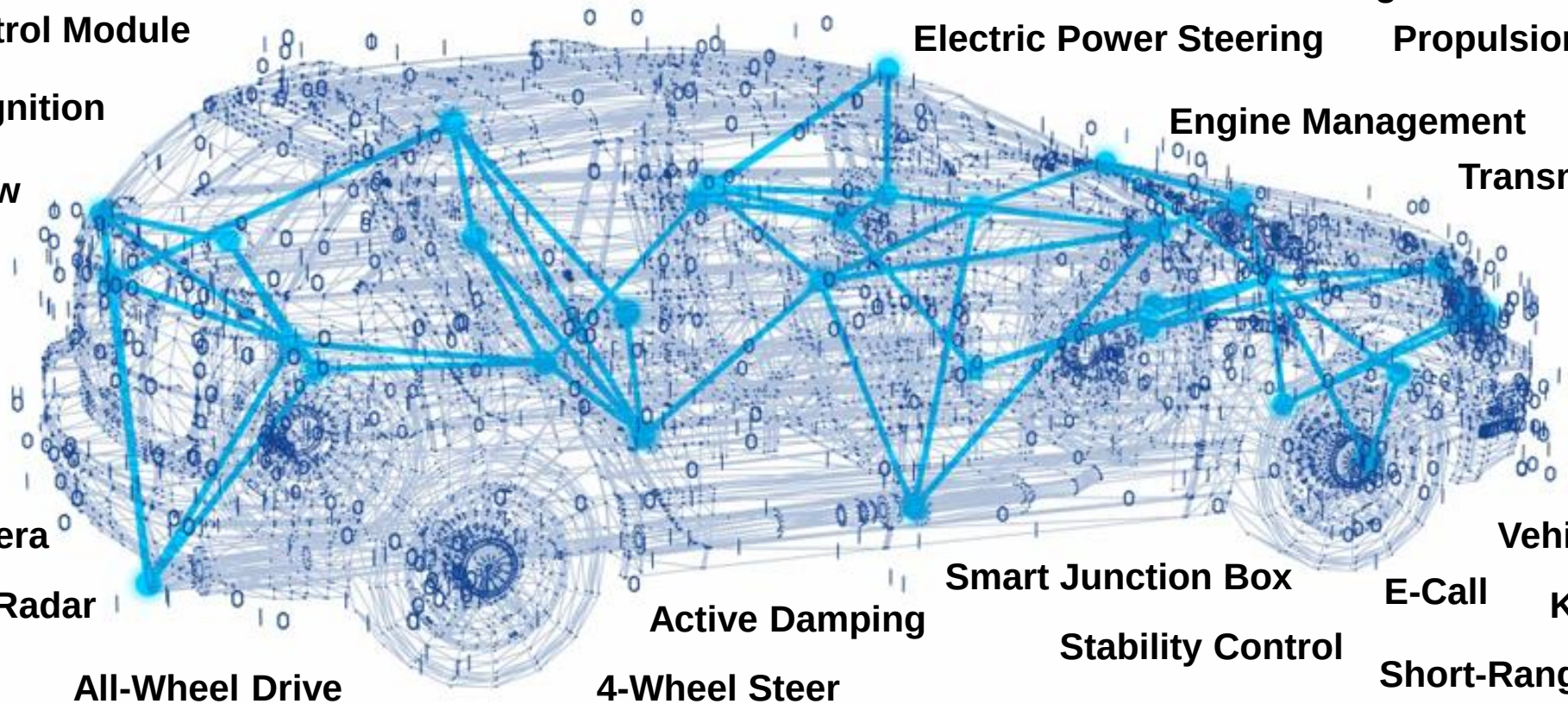
# MathWorks Vision for Systematic Verification and Validation

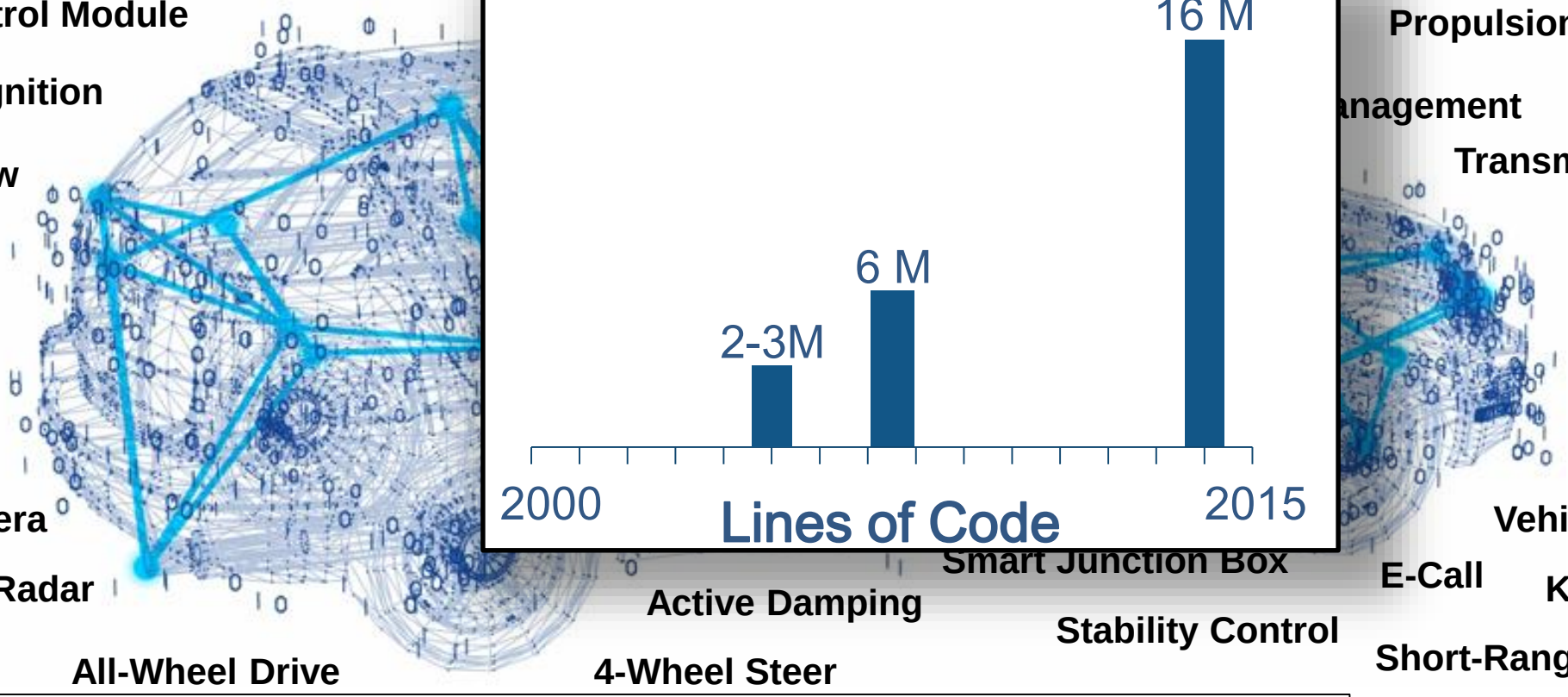
**Bill Aldrich**
**Senior Development Manager**
**Simulink Verification and Validation, Simulink Design Verifier**

# Growing Complexity of Automotive Controls




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

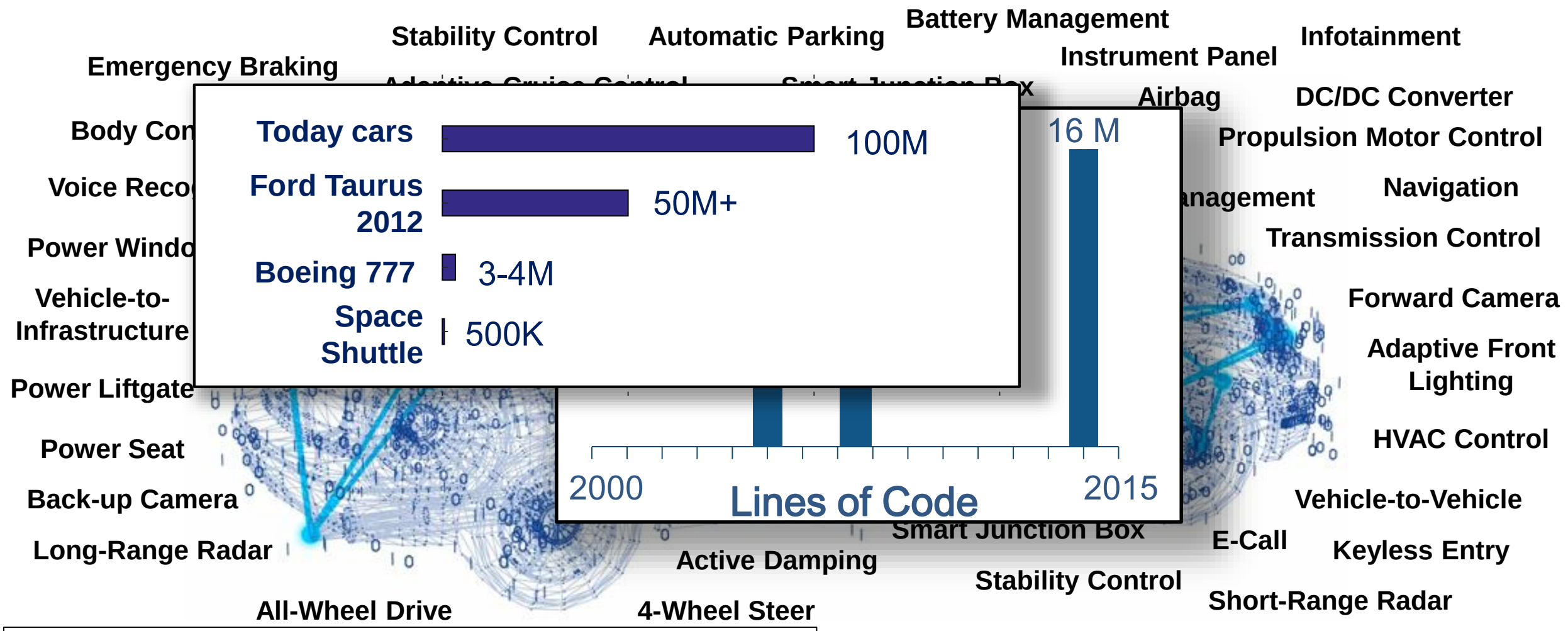
# Growing Complexity of Automotive Controls




Emergency Braking
Stability Control
Automatic Parking
Battery Management
Infotainment
Instrument Panel
Adaptive Cruise Control
Smart Junction Box
Airbag
DC/DC Converter
Body Control Module
Propulsion Motor Control
Voice Recognition
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
Long-Range Radar
Keyless Entry
Active Damping
Stability Control
Short-Range Radar
All-Wheel Drive
4-Wheel Steer

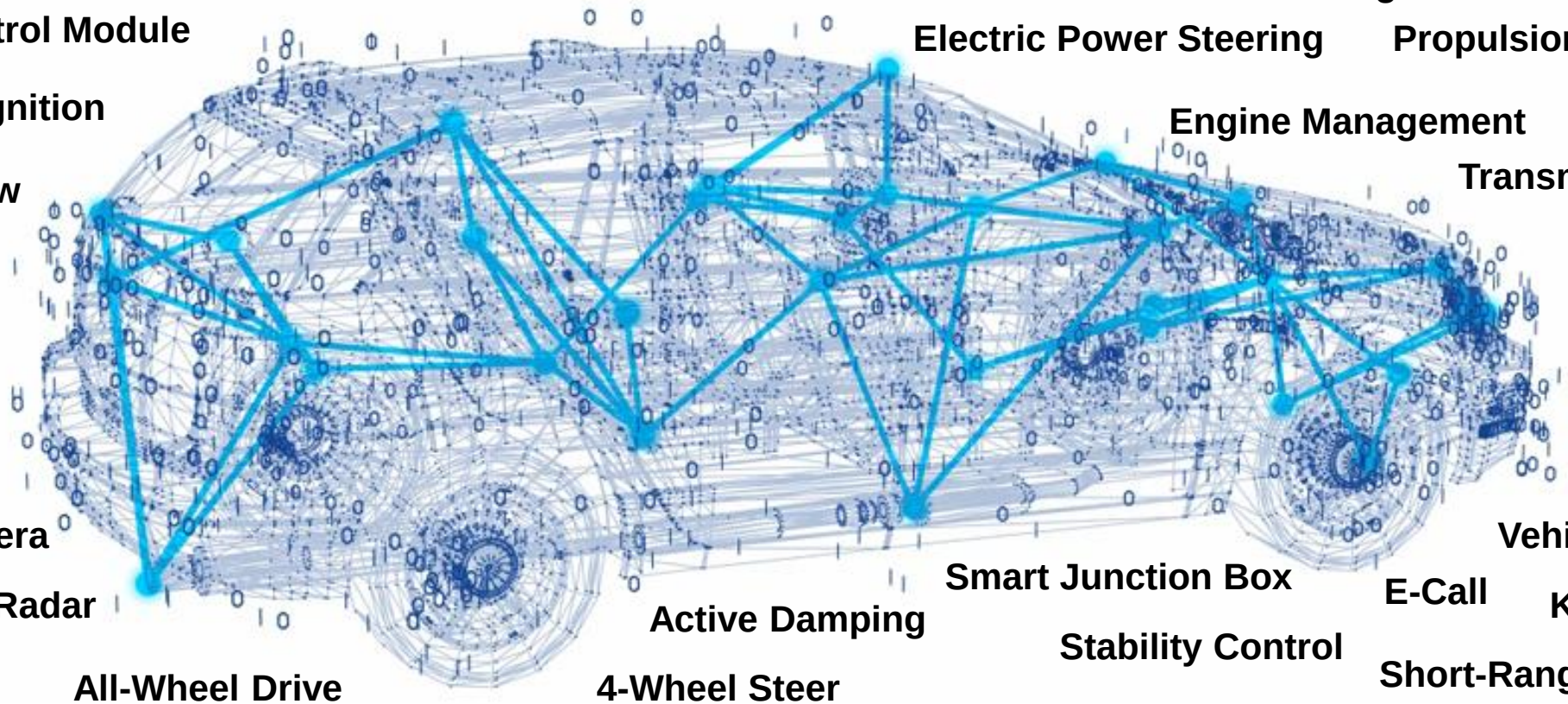
Siemens, "Ford Motor Company Case Study," Siemens PLM Software, 2014
McKendrick, J. "Cars become 'datacenters on wheels', carmakers become software companies," ZDJNet, 2013

# Growing Complexity of Automotive Controls




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

# Growing Complexity of Automotive Controls

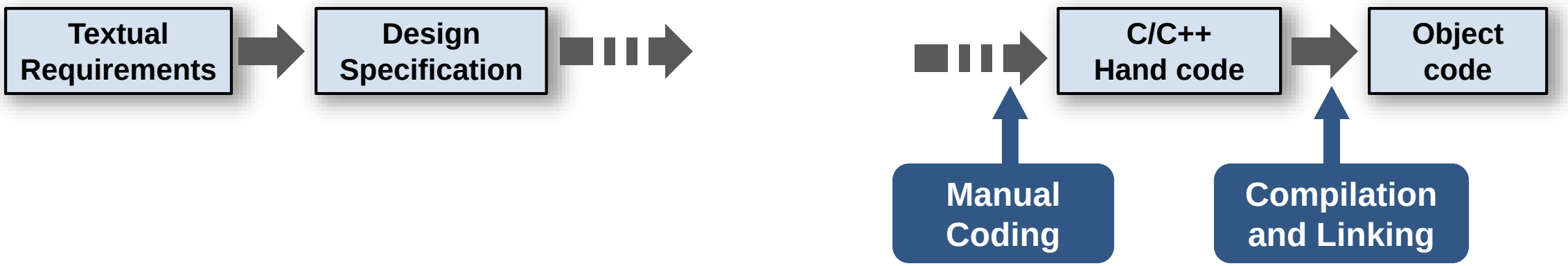



Source of graphic: http://360.here.com/2013/11/28/putting-firmly-drivers-seat/

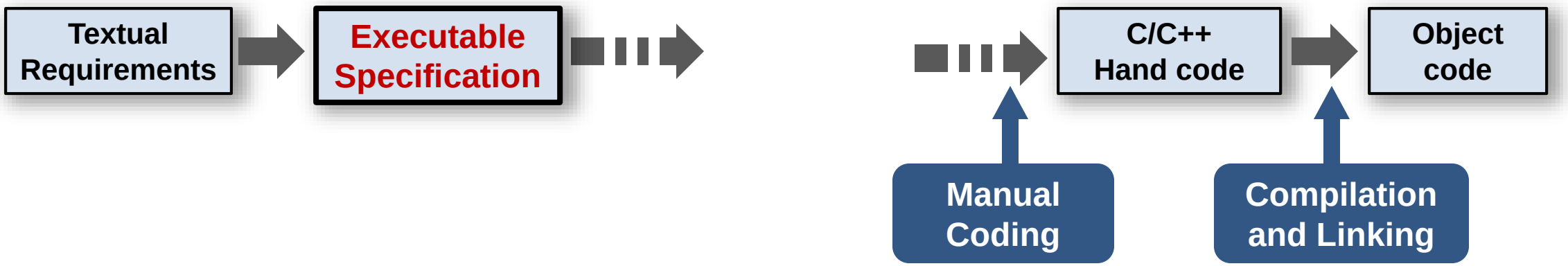
# Development Challenges

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

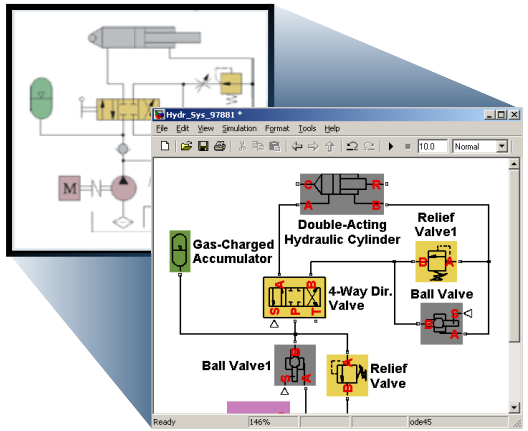
# Traditional Development Process



# Models for Specification

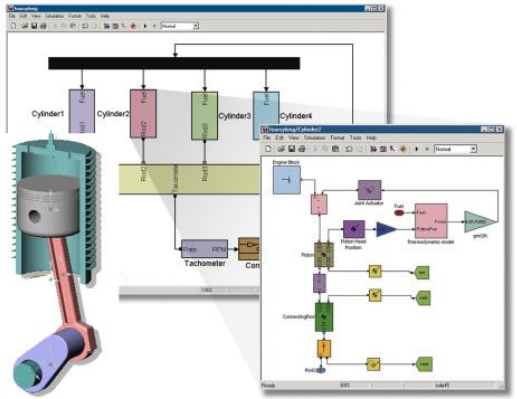


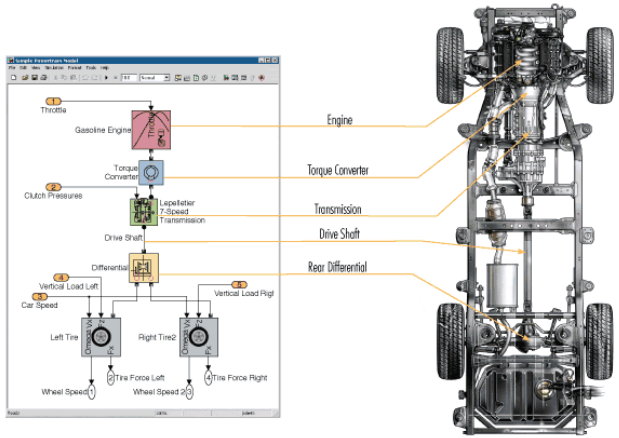
# Model Abstraction – Work at an appropriate level of detail




Simscape Fluids

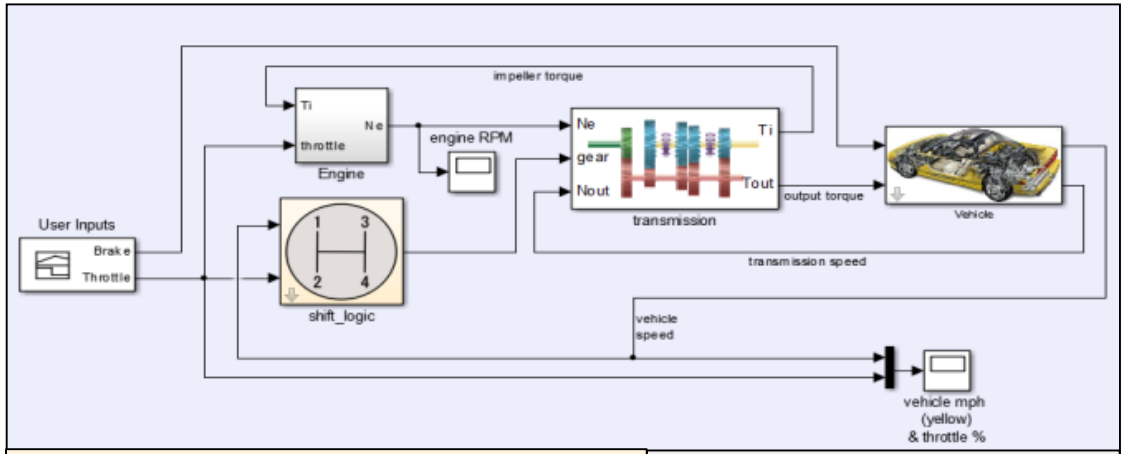
Simscape Multibody

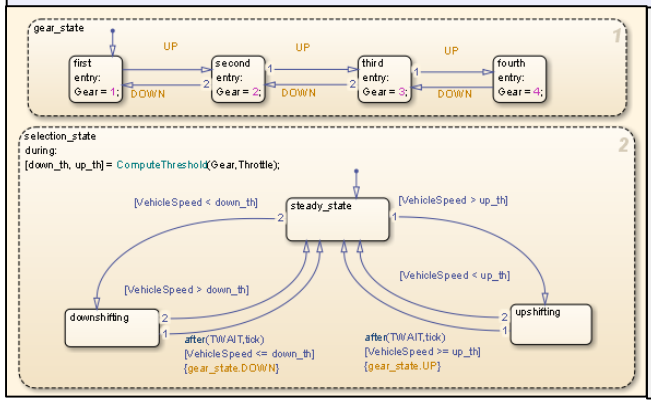





Simscape Driveline

Simulink



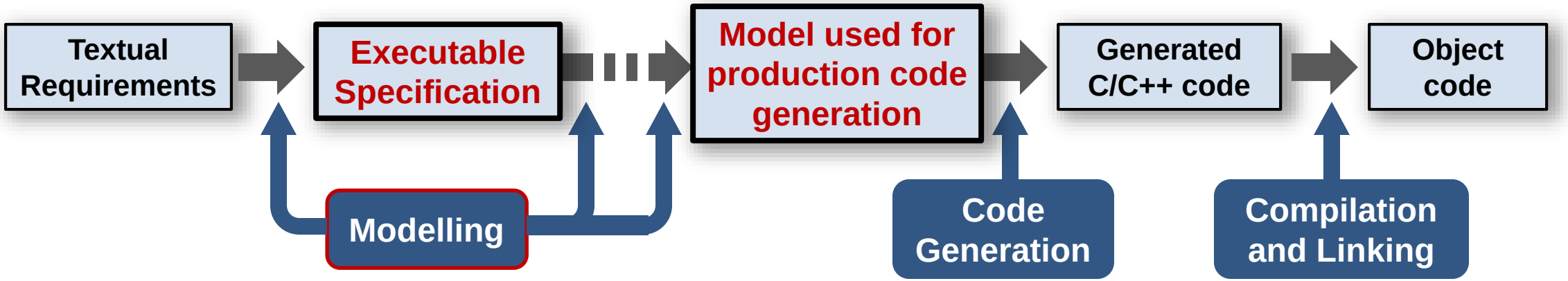




Stateflow

```
% 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;
```

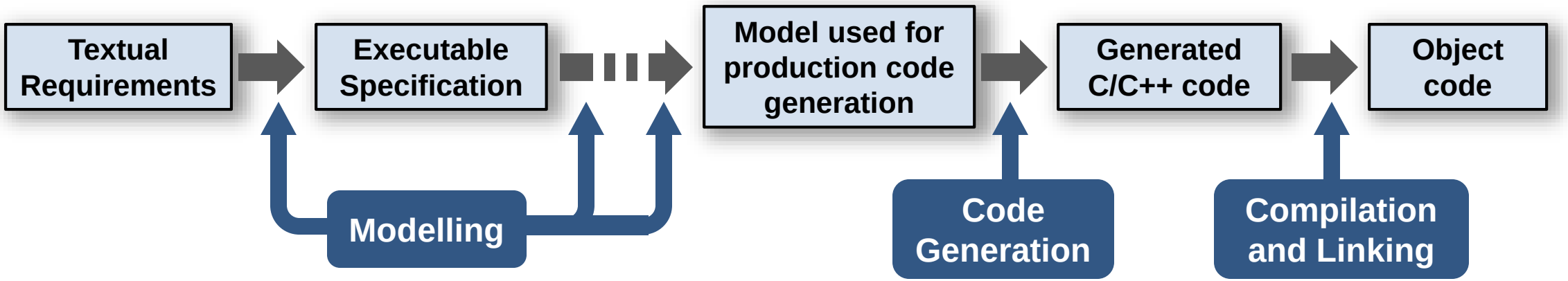
MATLAB

# Complete Model Based Design Workflow, Concept to Code

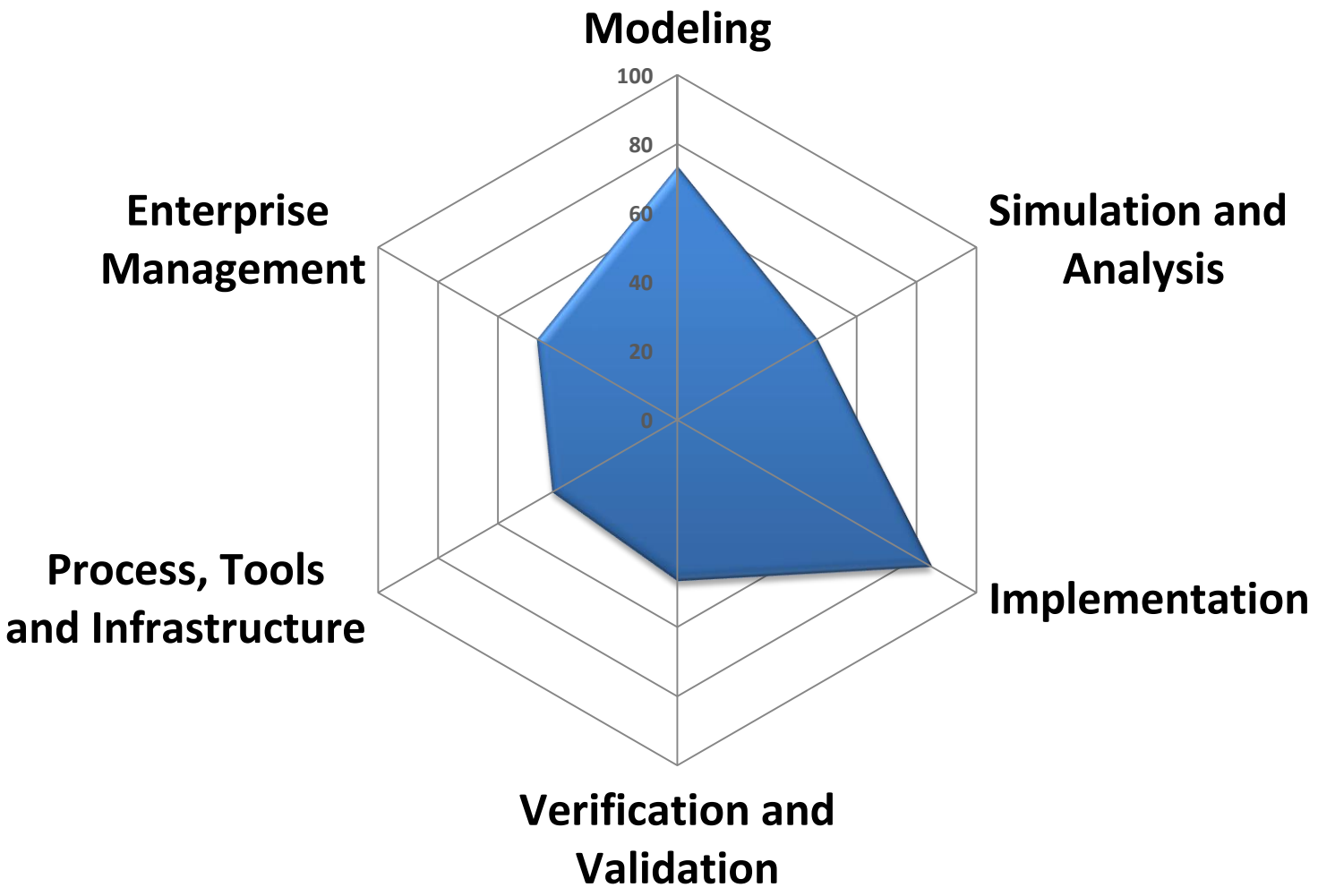


# Complete Model Based Design Workflow, Concept to Code

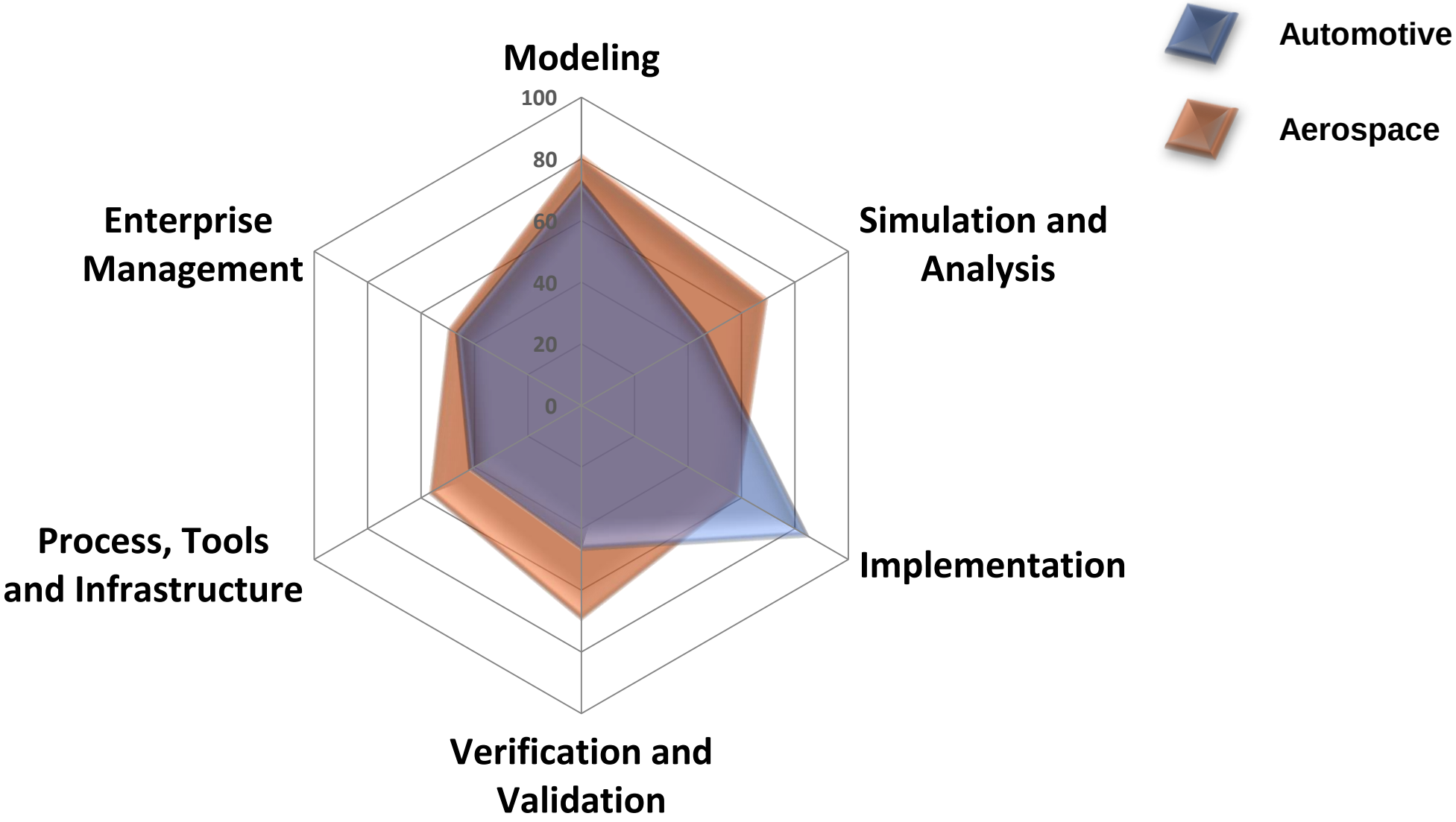
## How do you ensure correctness?



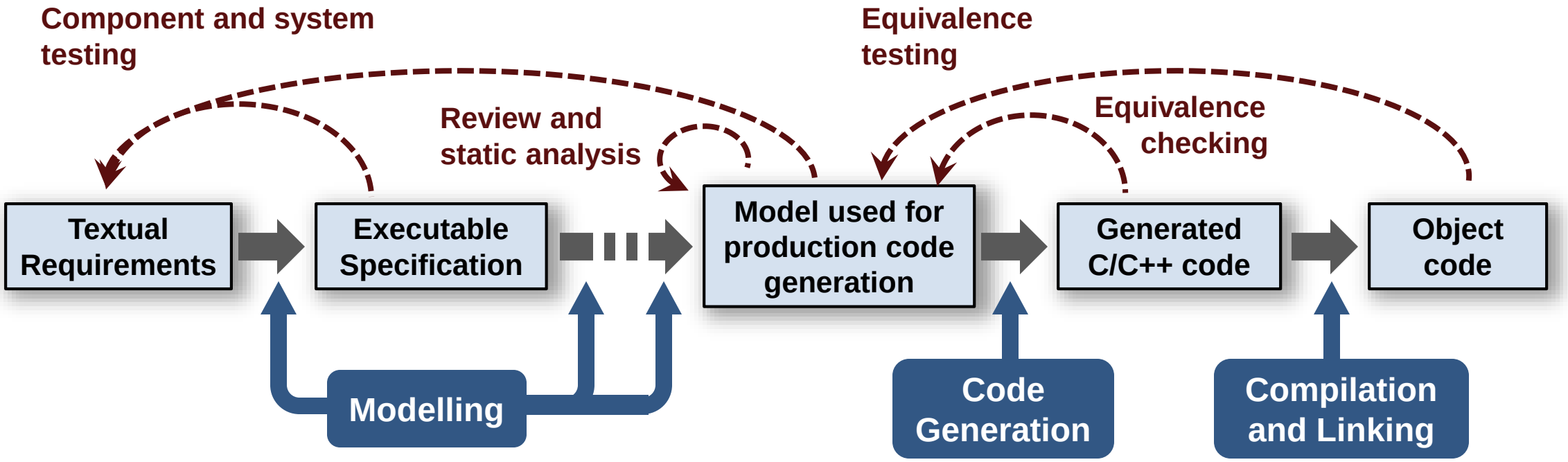
# Model-Based Design Maturity, Automotive Industry



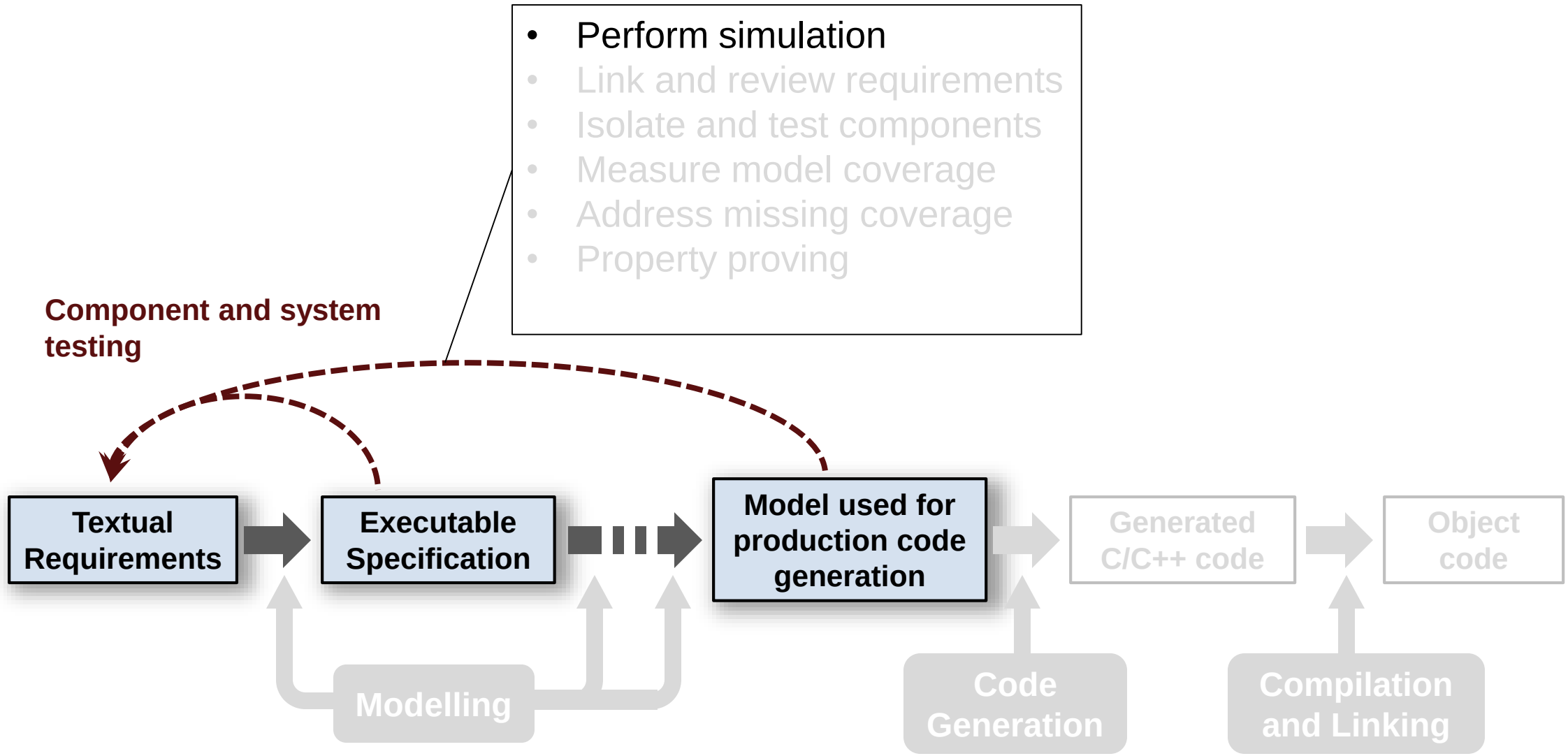
# Model-Based Design Maturity, Automotive and Aerospace



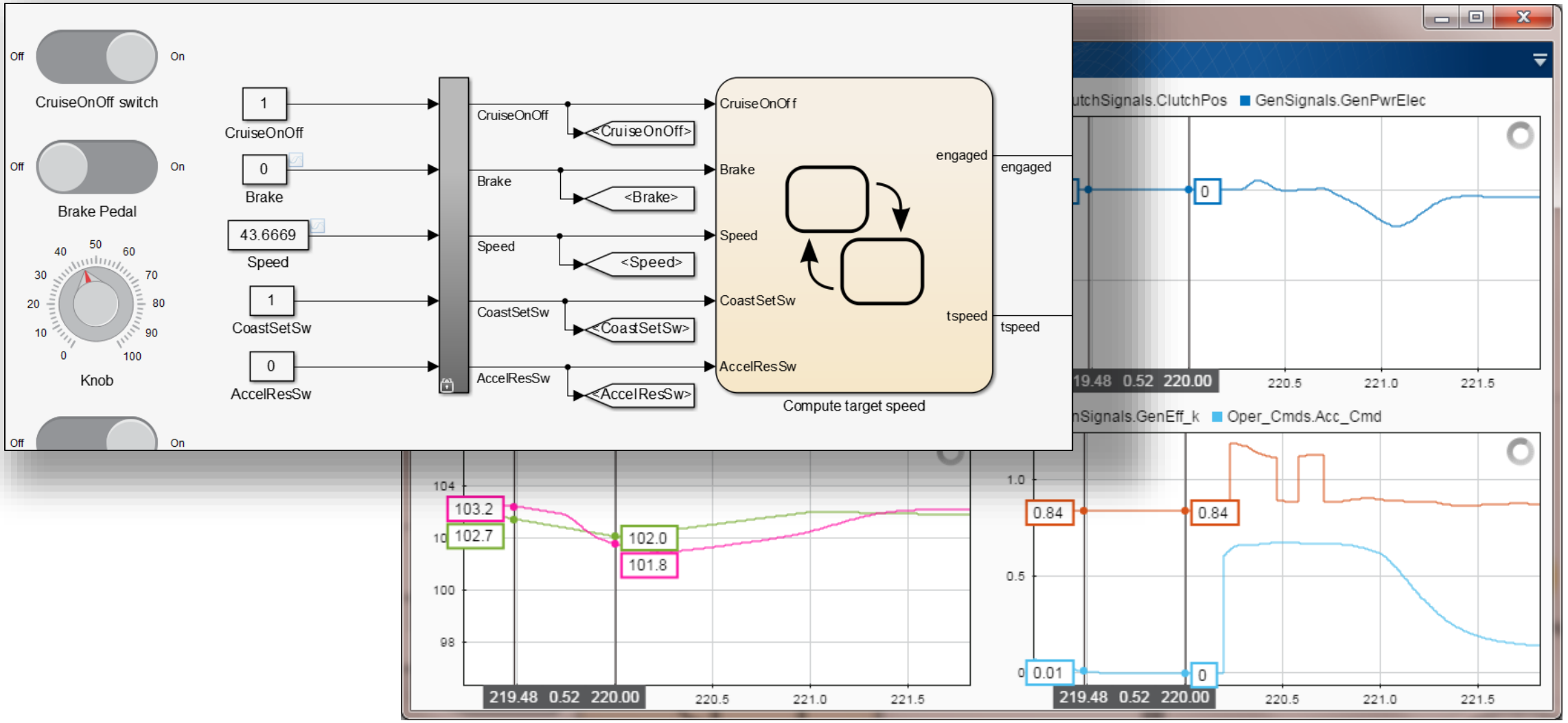
# Model Based Design Verification Workflow



# Model Based Design Verification Workflow

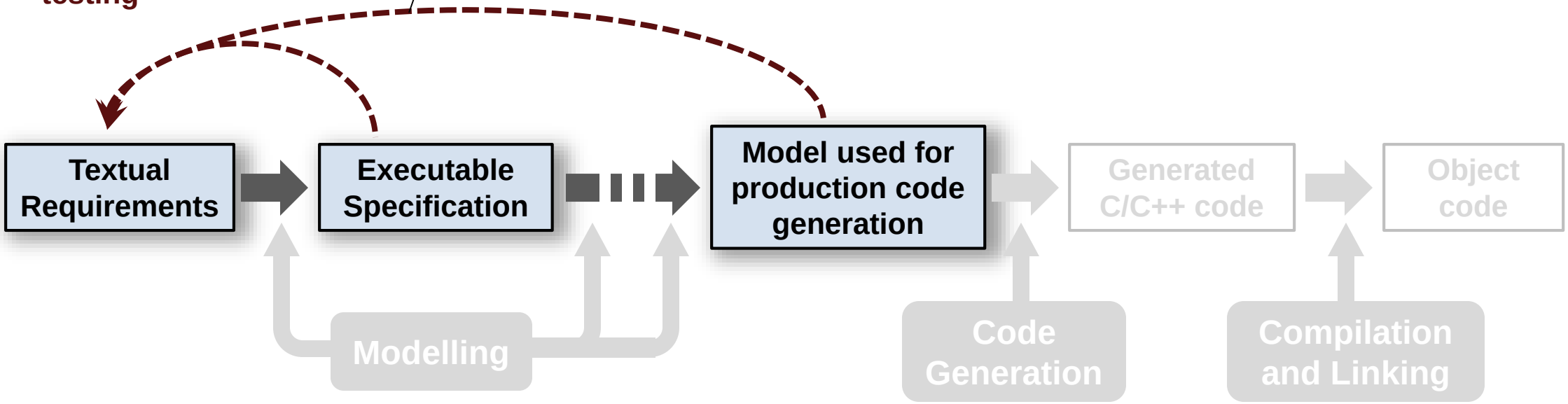


# Ad-Hoc Simulation: Explore Behavior Virtually



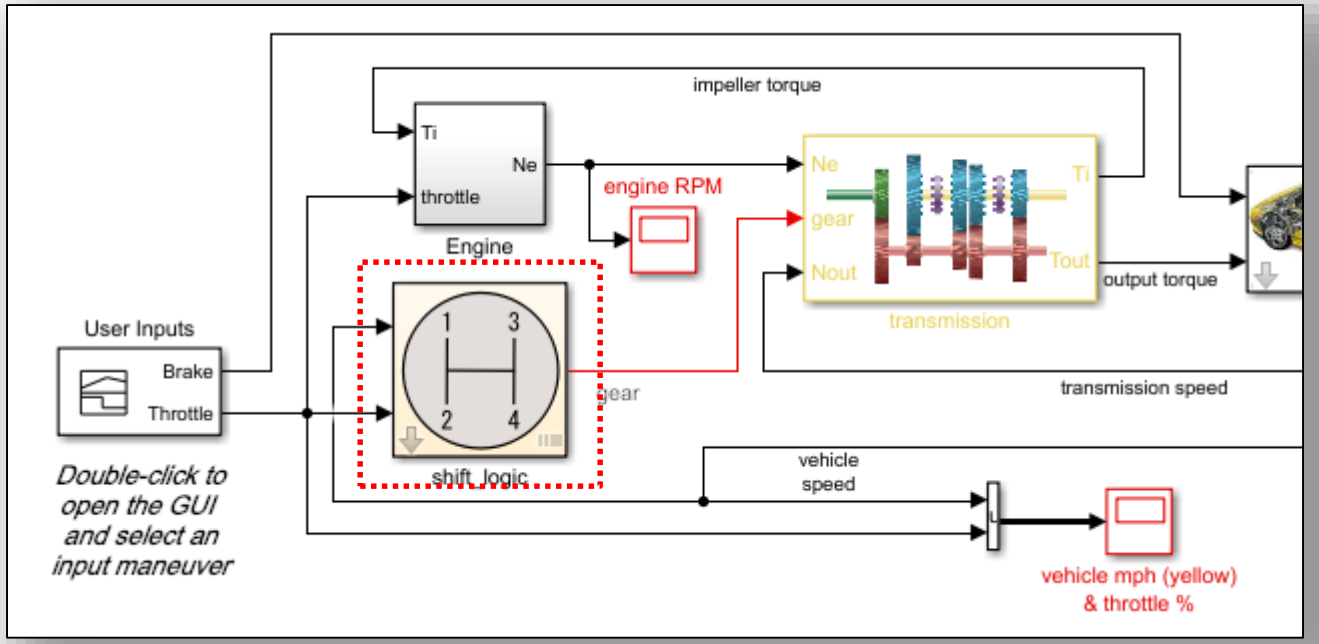
# Model Based Design Verification Workflow

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



# Test Harnesses

**From <u>any</u> subsystem …**

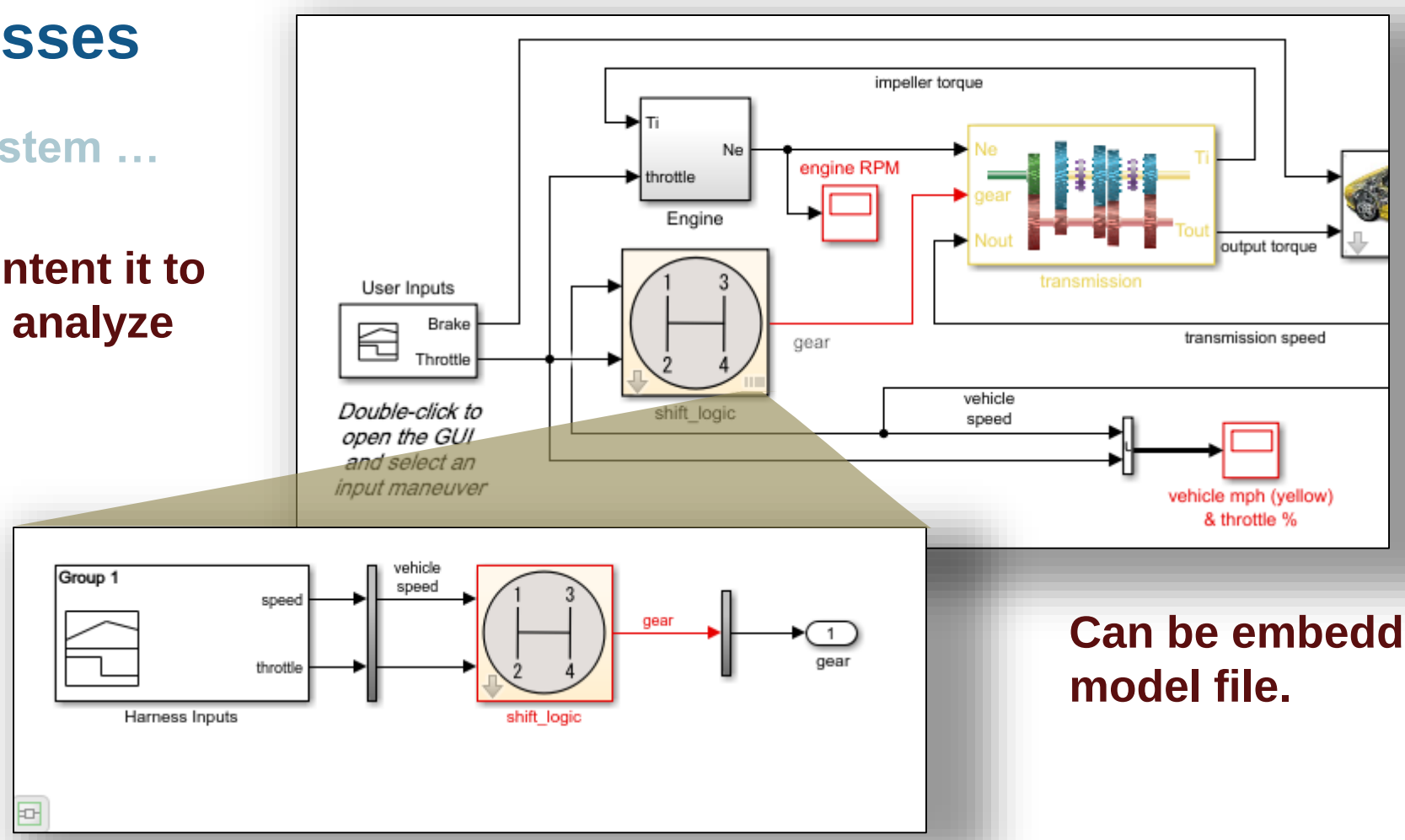


# Test Harnesses

From any subsystem …

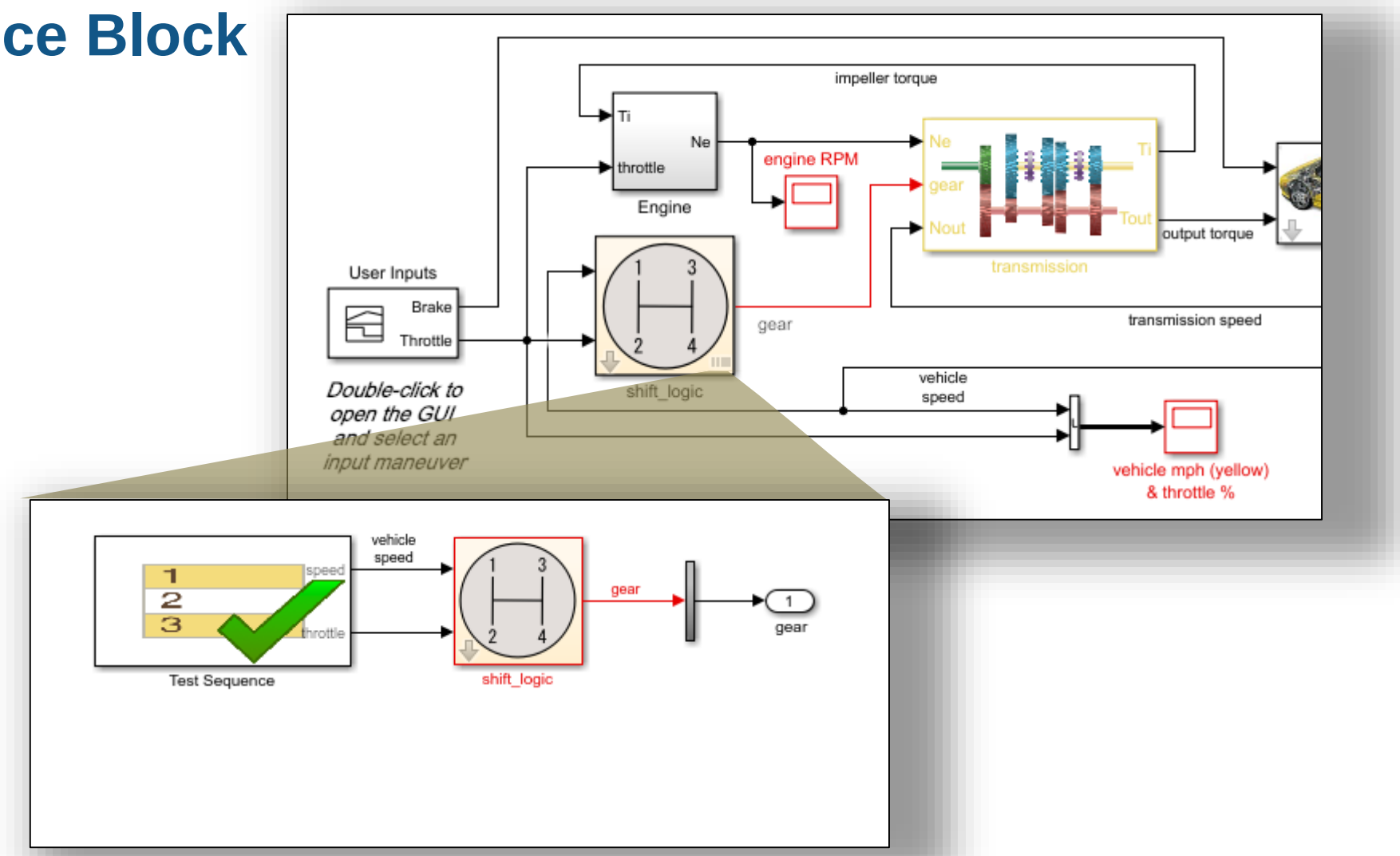
**Isolate it with content it to drive inputs and analyze outputs**

**Simulate independently**



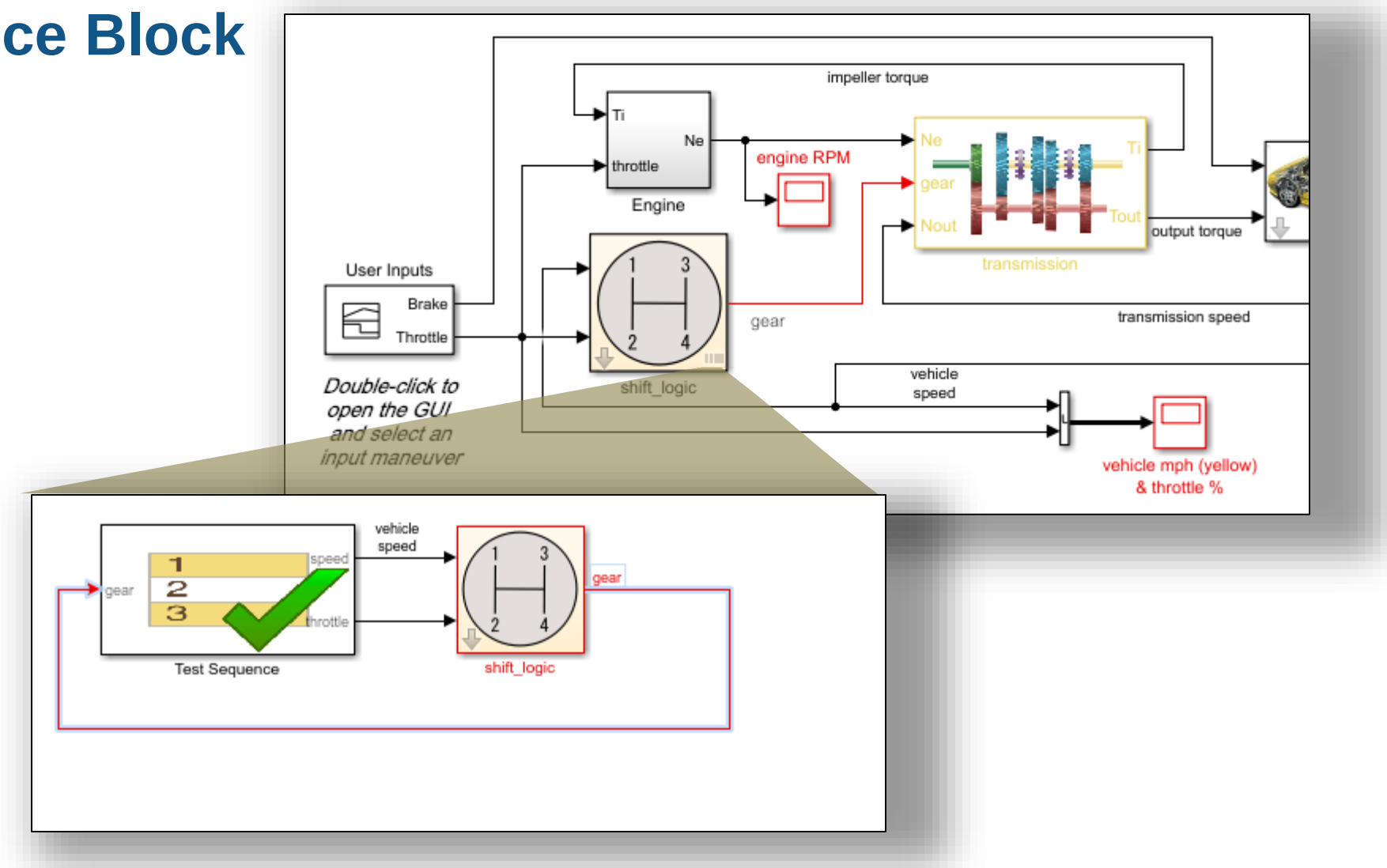

**Can be embedded in design model file.**

# Test Sequence Block



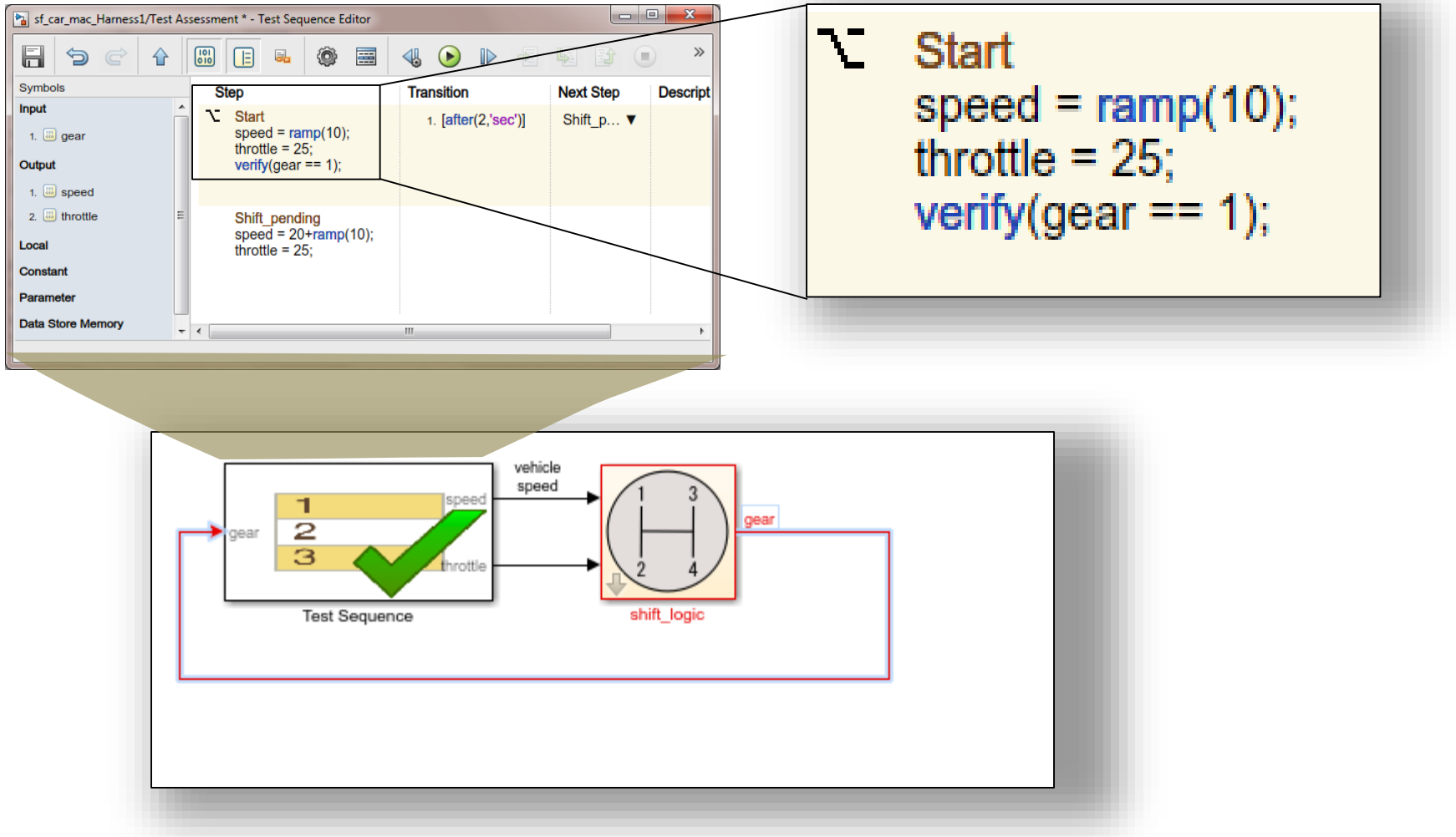

**A test sequence block can drive inputs**

# Test Sequence Block

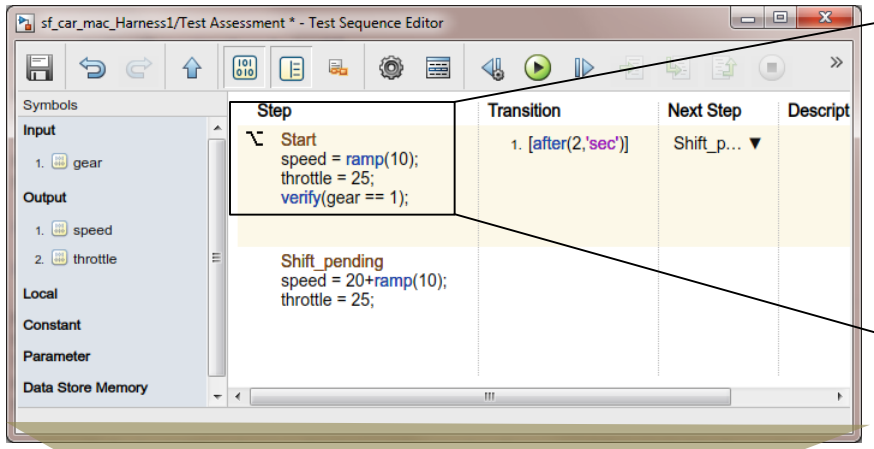


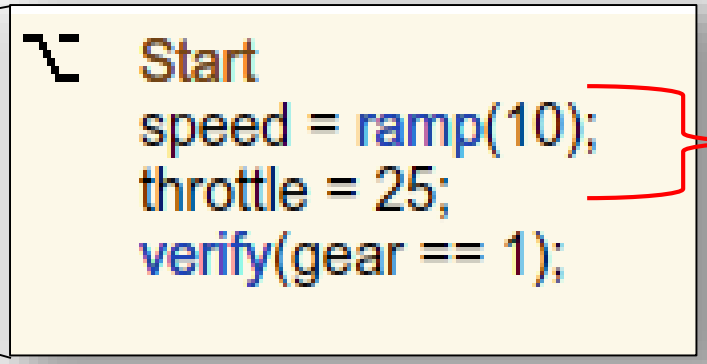

**A test sequence block can drive inputs and asses outputs**

# Test Sequence Block Syntax

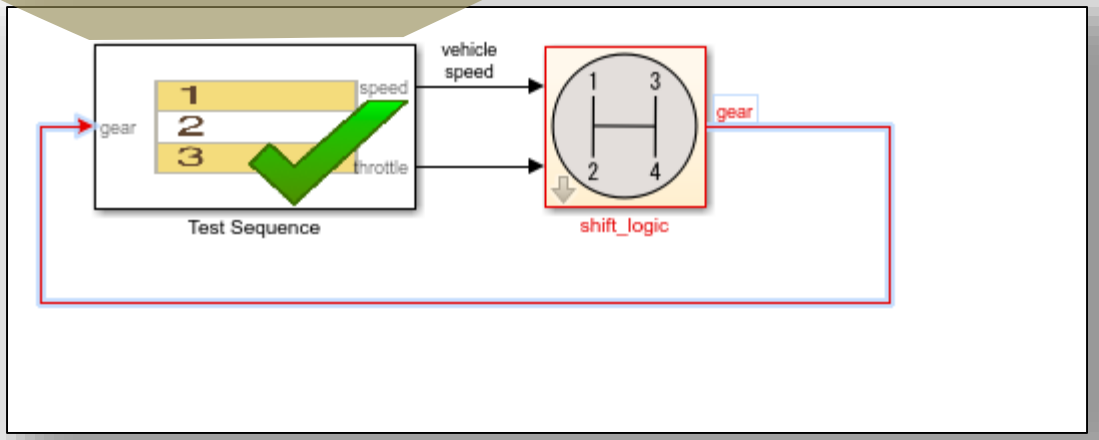


# Test Sequence Block Syntax

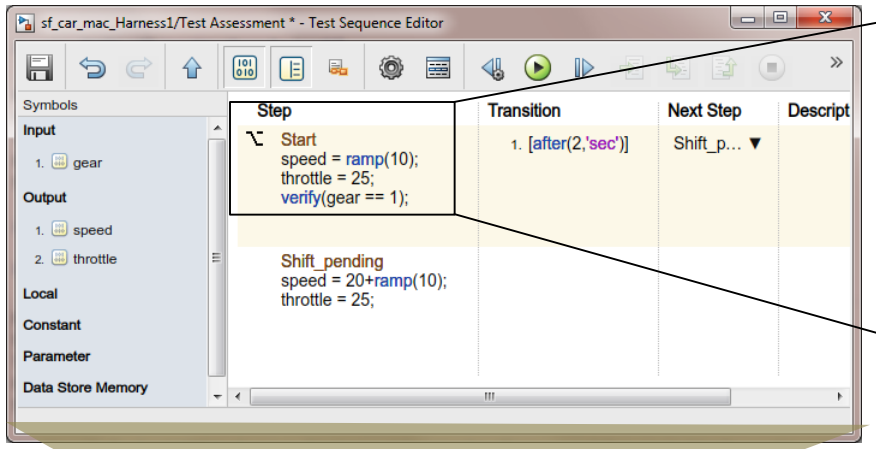


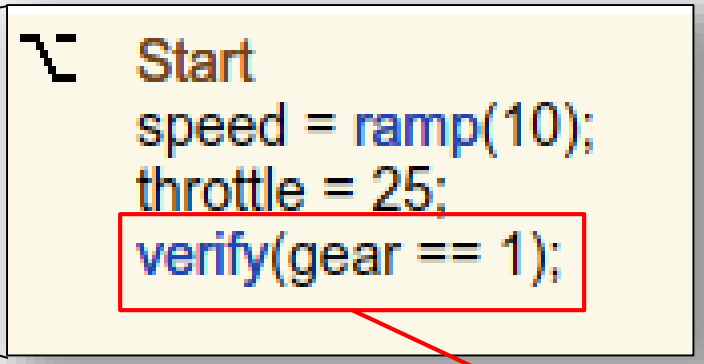



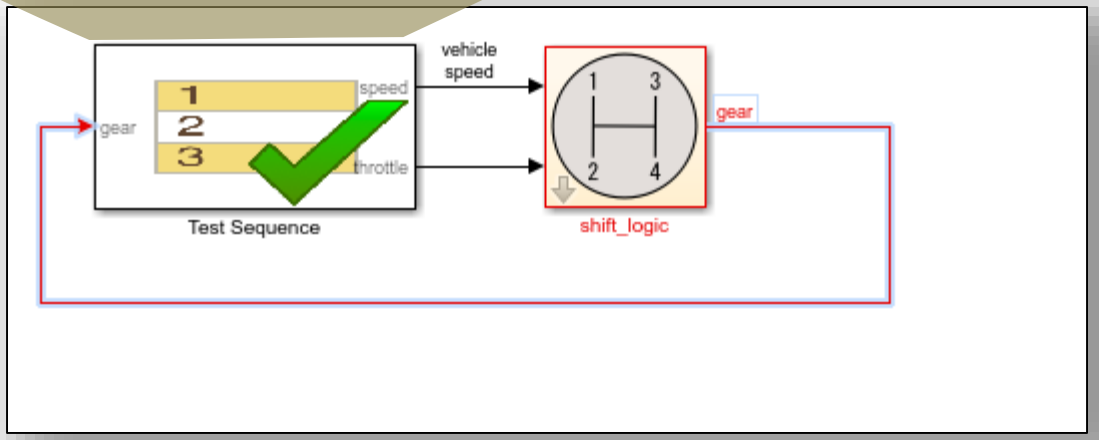

**Define Inputs**

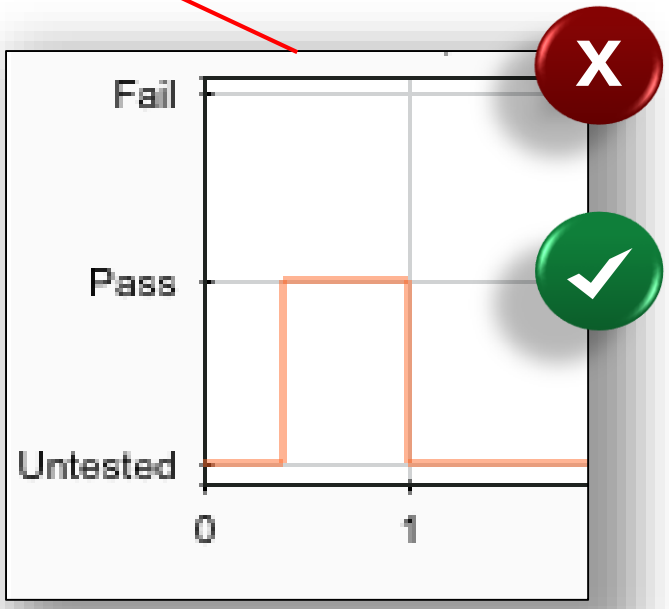


# Defining Pass/Fail Criteria



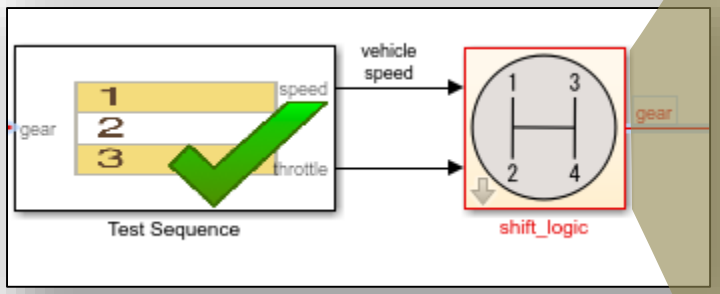



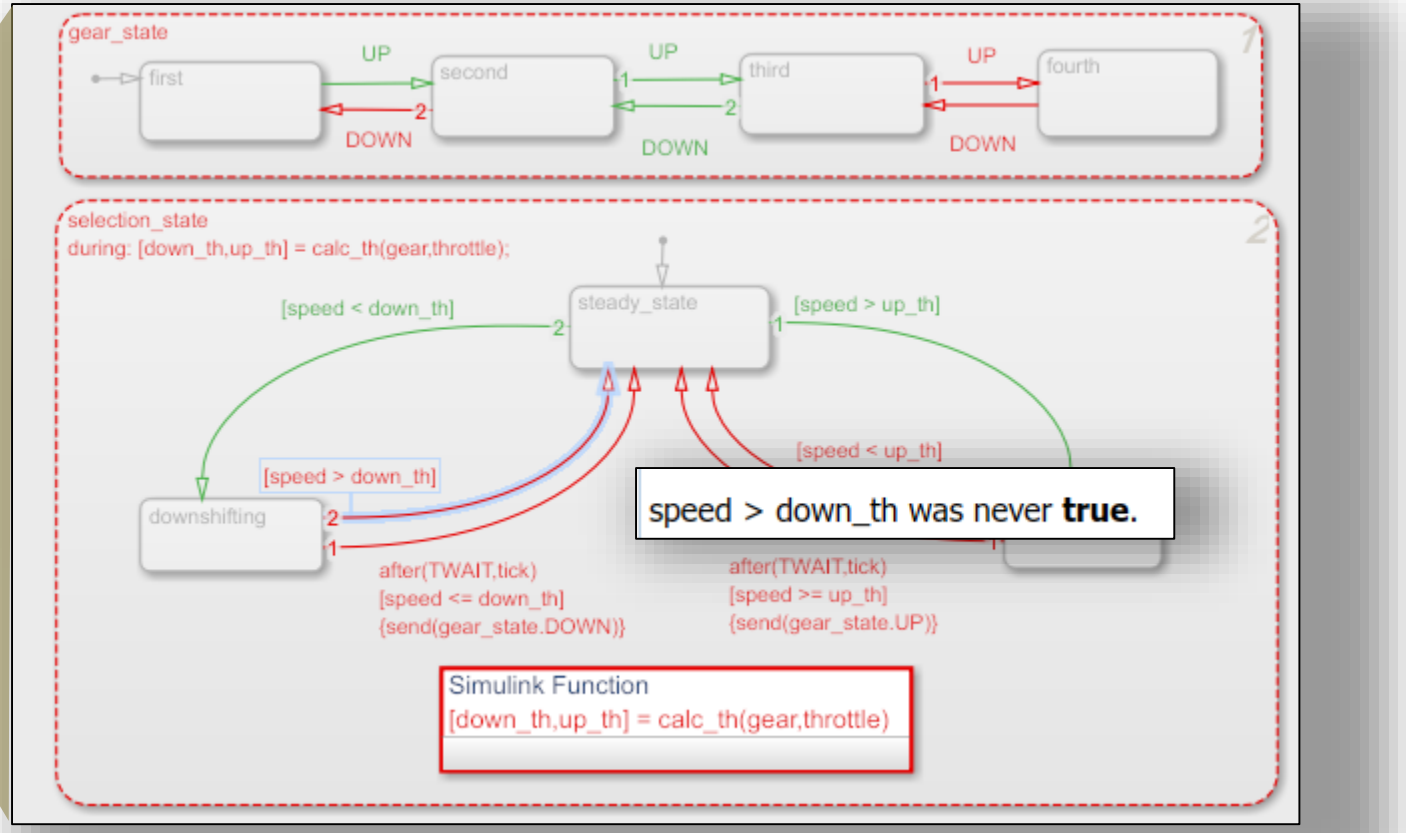



# Model Coverage

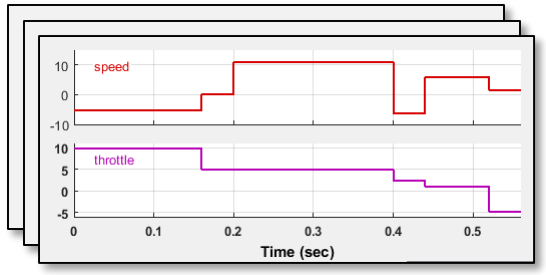
**Identify testing gaps:**

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

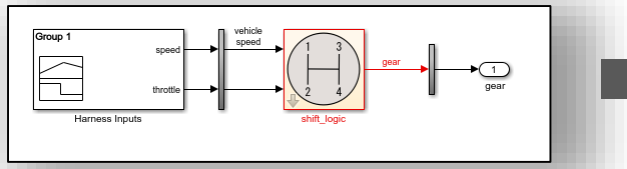


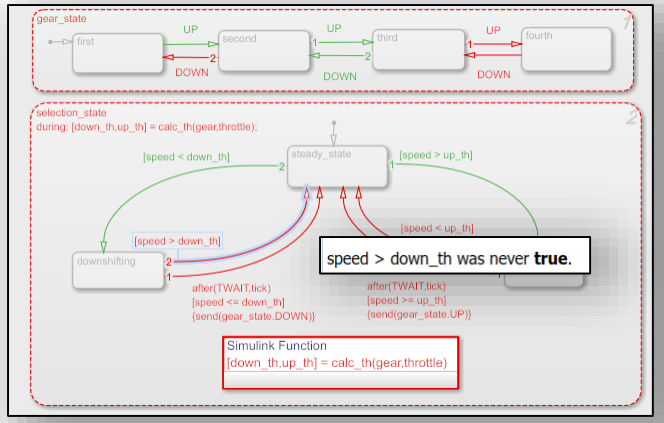
# Addressing Missing Coverage



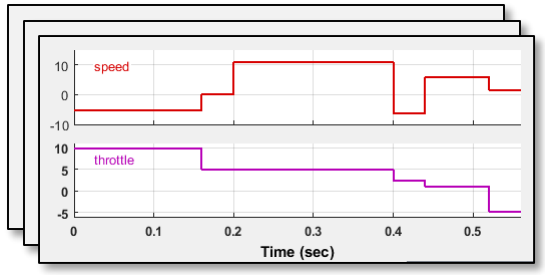

Test Cases



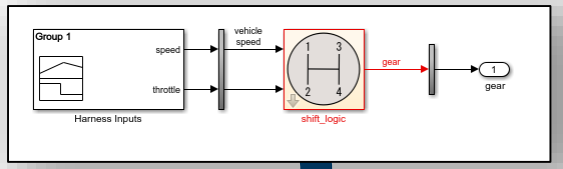

Partial Coverage



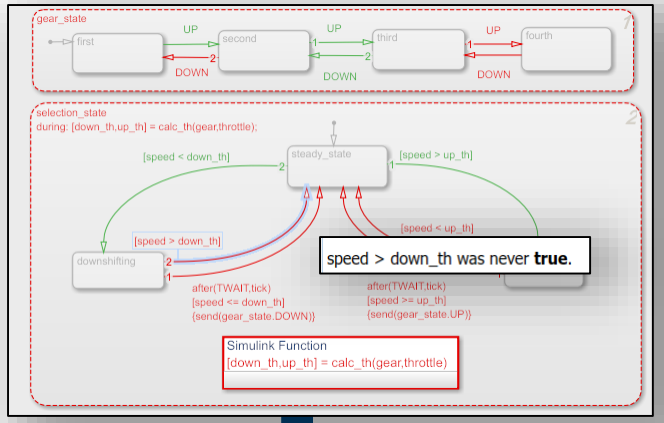
# Addressing Missing Coverage




Test Cases




Partial Coverage

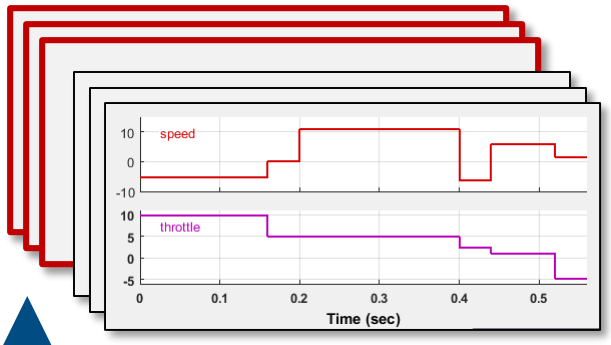



Test Generator

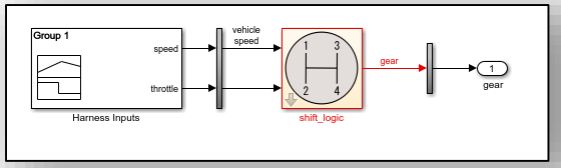
Simulink Design Verifier

# Addressing Missing Coverage

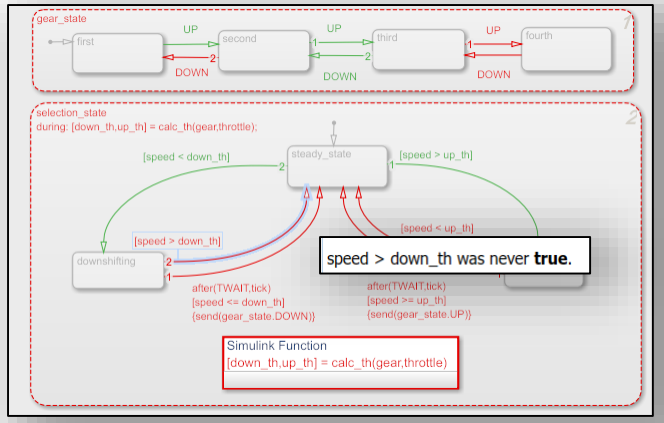
## New Test Cases




## Test Cases




## Partial Coverage

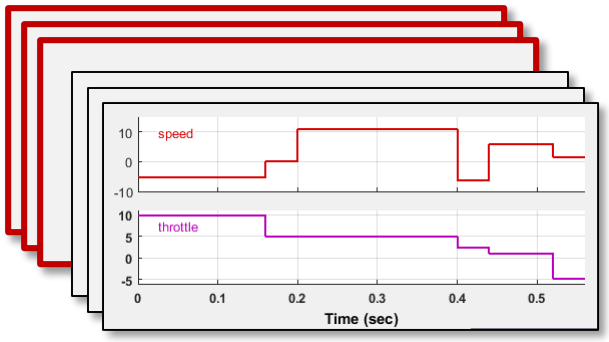



**Test Generator**

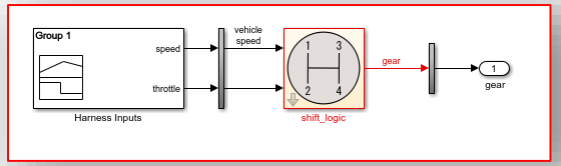
## Simulink Design Verifier

# Addressing Missing Coverage

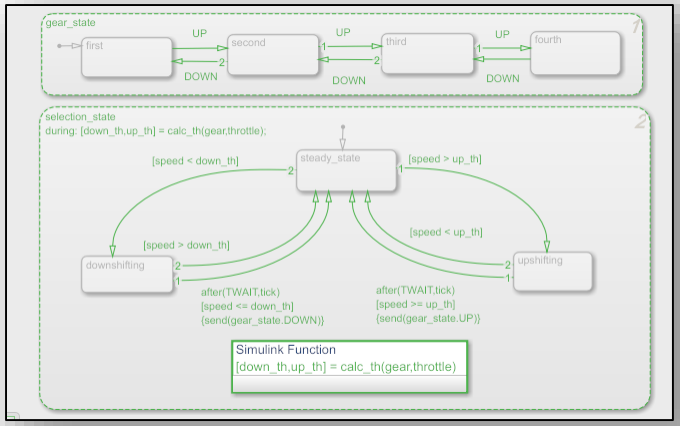
## New Test Cases




Test Cases

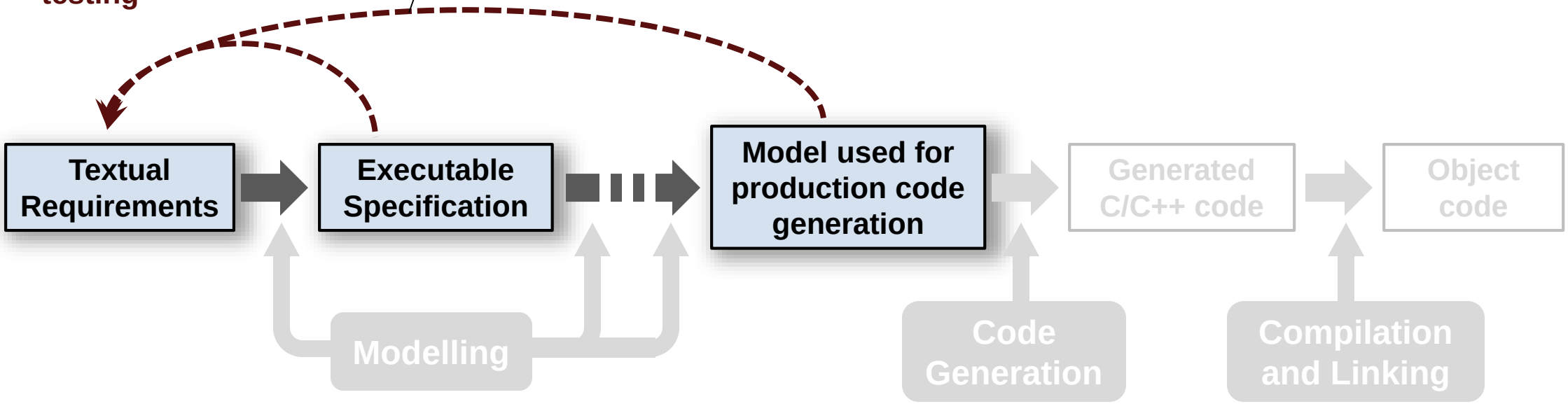



## Full Coverage



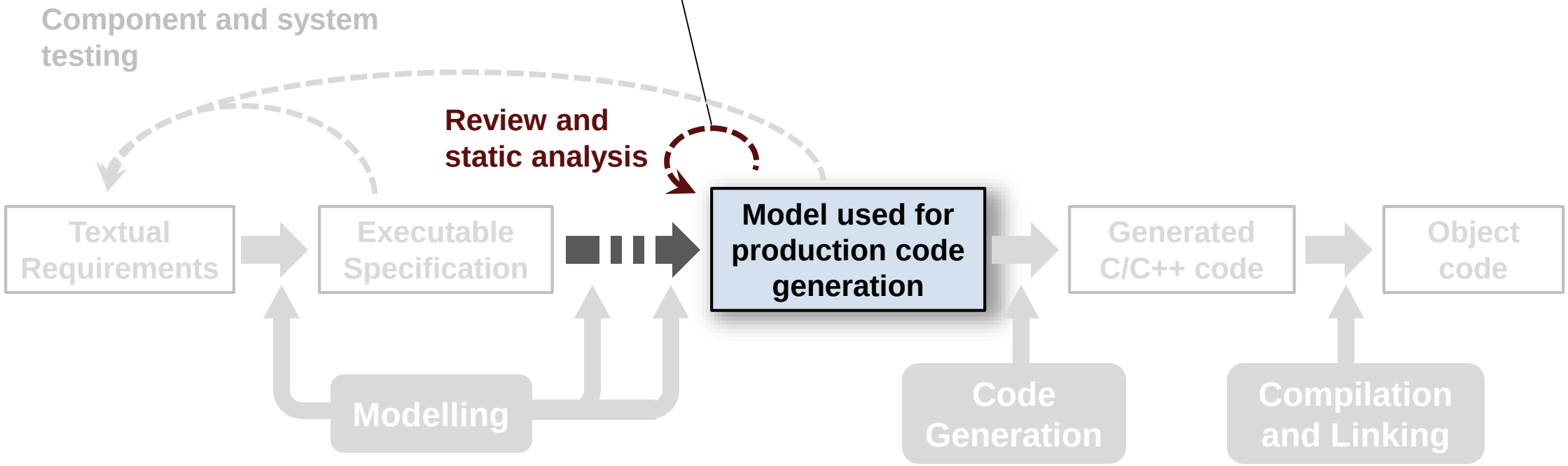
# Model Based Design Verification Workflow

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

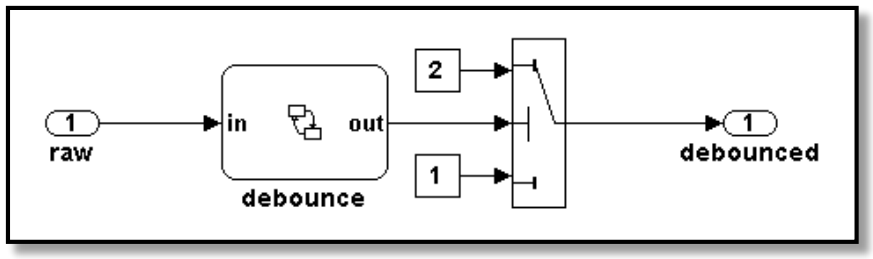


# Model Based Design Verification Workflow

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

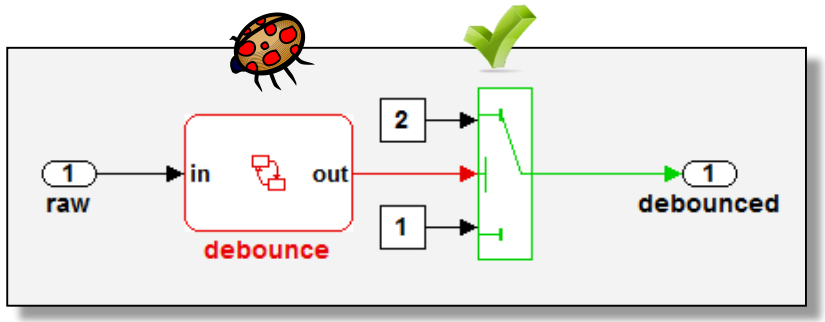


# Detecting Hidden Run-Time Design Errors




Design Model

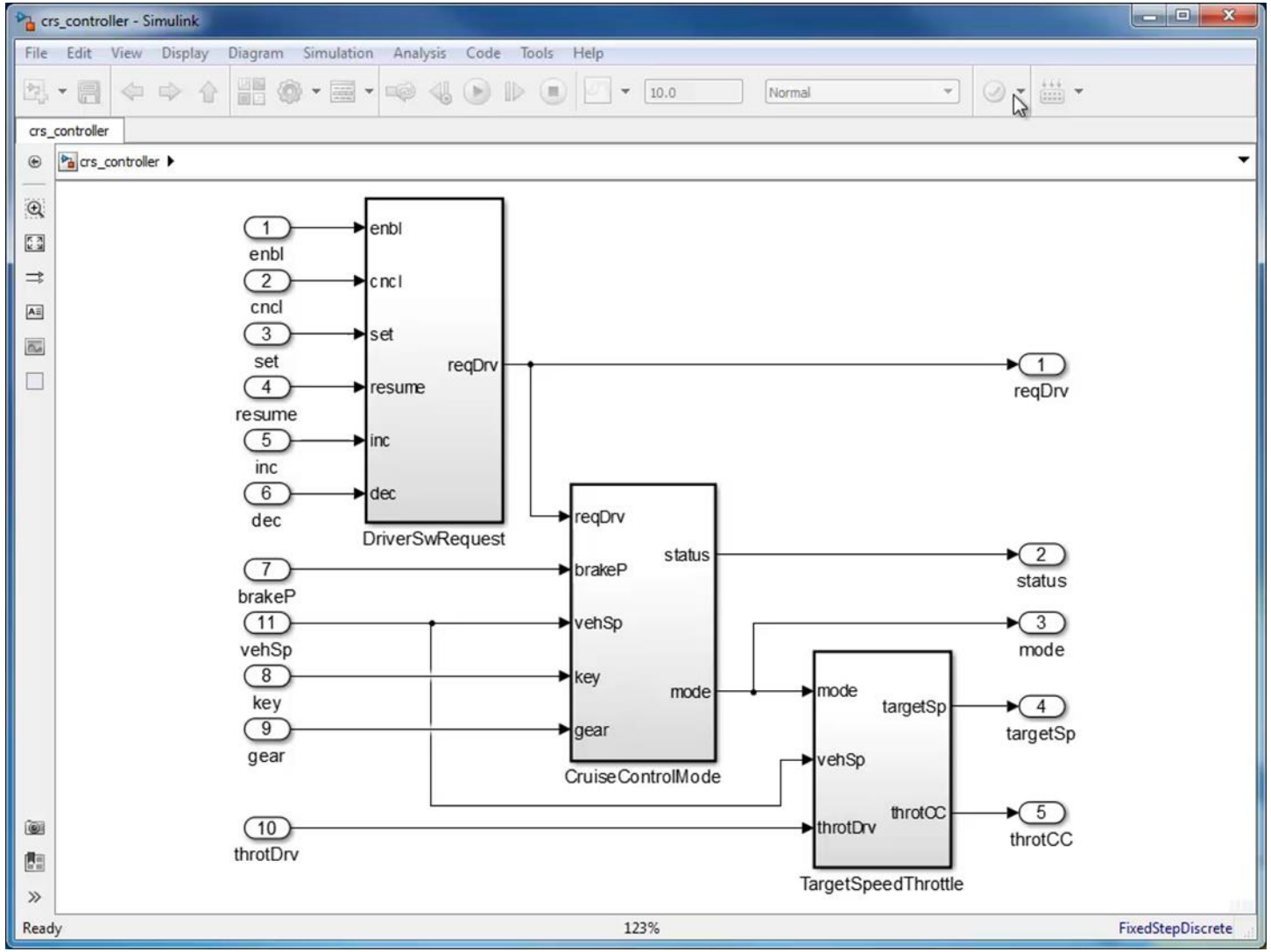
Design error detection



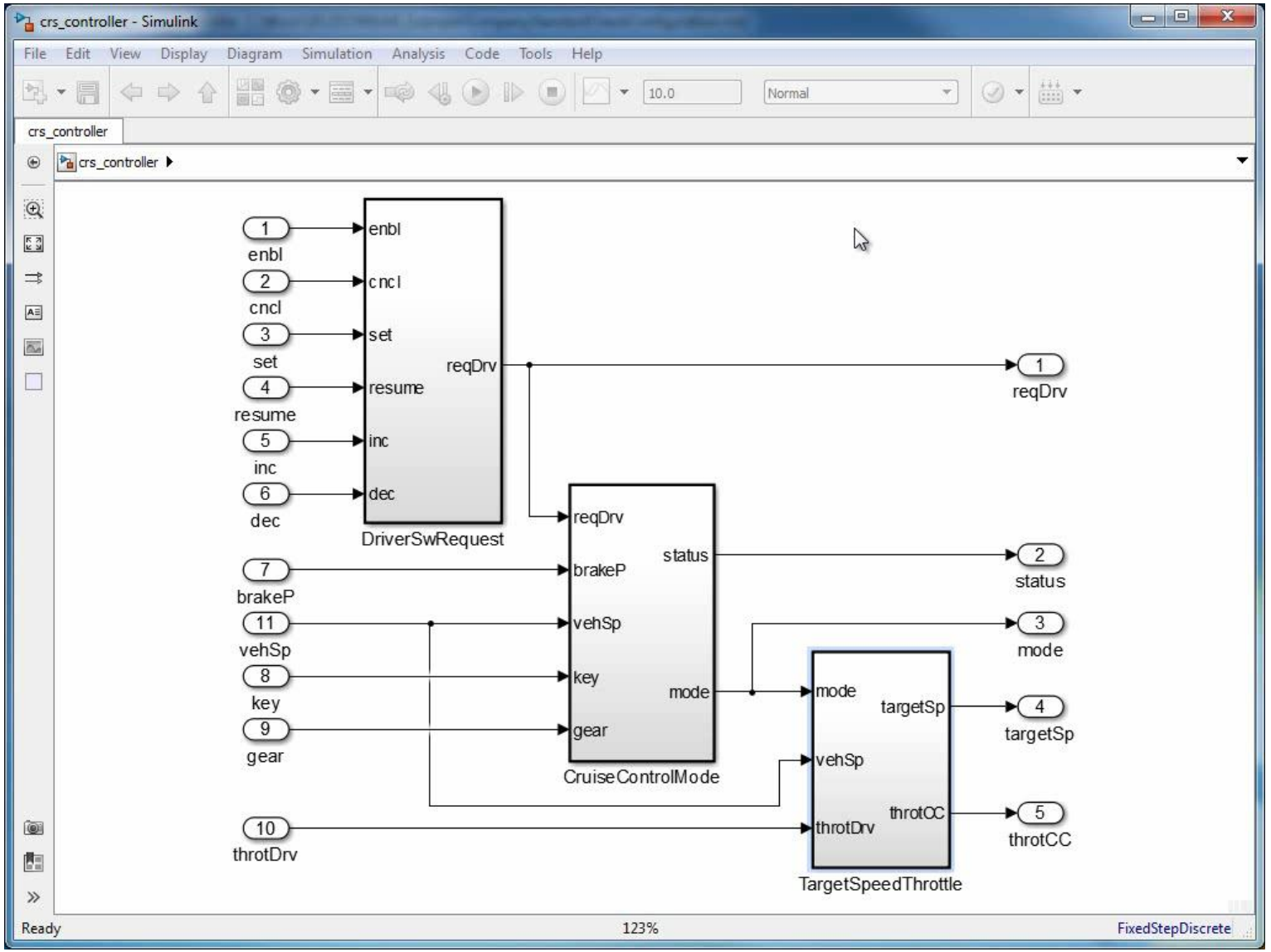

Highlighted Model

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

# Detecting Hidden Run-Time Design Errors

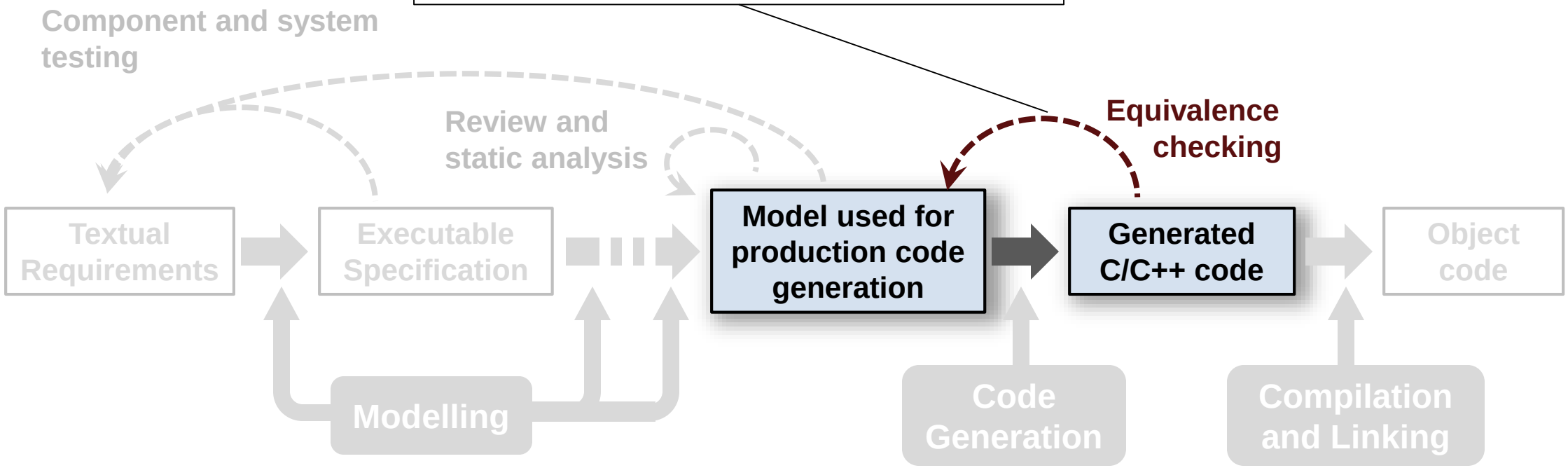


# Detecting Hidden Run-Time Design Errors

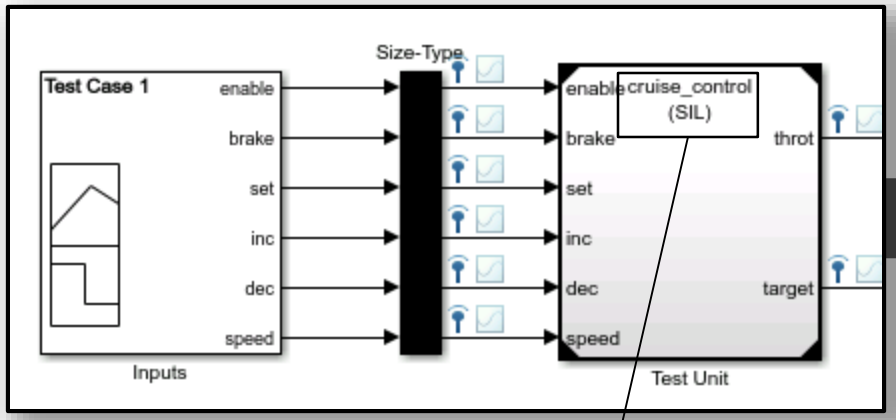


# Model Based Design Verification Workflow

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



# Coverage for Generated Code (R2016a)



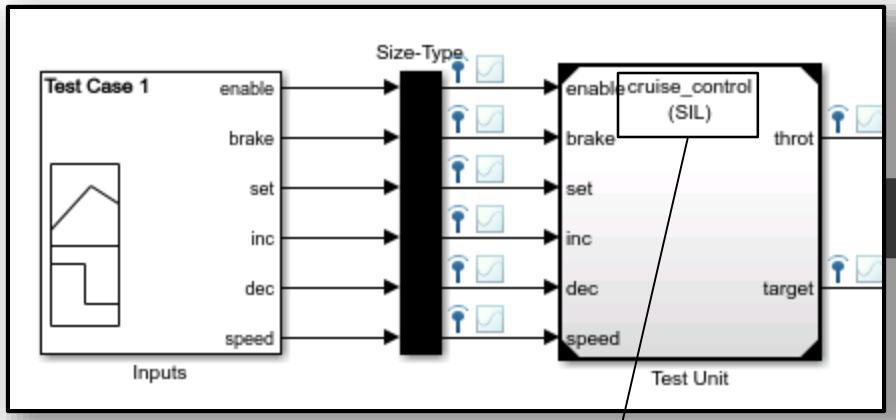

**cruise_control (SIL)**

**Press Play**

```
77    if (rtb_ActiveControl) {
78      /* Sum: '<S2>/Sum' incorporates:
79       *  DiscreteIntegrator: '<S2>/Discrete-Time Integrator'
80       *  Gain: '<S2>/Kp'
81       *  Gain: '<S2>/Kp1'
82       */
83      *rty_throt = 0.02 * rtb_Switch2 + 0.01 *
84        localDW->DiscreteTimeIntegrator_DSTATE;
85
86      /* Update for DiscreteIntegrator: '<S2>/Discrete-Time Integrator' *
87      localDW->DiscreteTimeIntegrator_DSTATE += 0.01 * rtb_Switch2;
88      if (localDW->DiscreteTimeIntegrator_DSTATE >= 5.0) {
89        localDW->DiscreteTimeIntegrator_DSTATE = 5.0;
90      } else {
91        if (localDW->DiscreteTimeIntegrator_DSTATE <= -5.0) {
92          localDW->DiscreteTimeIntegrator_DSTATE = -5.0;
93        }
94      }
```

**Generated Code Coverage**

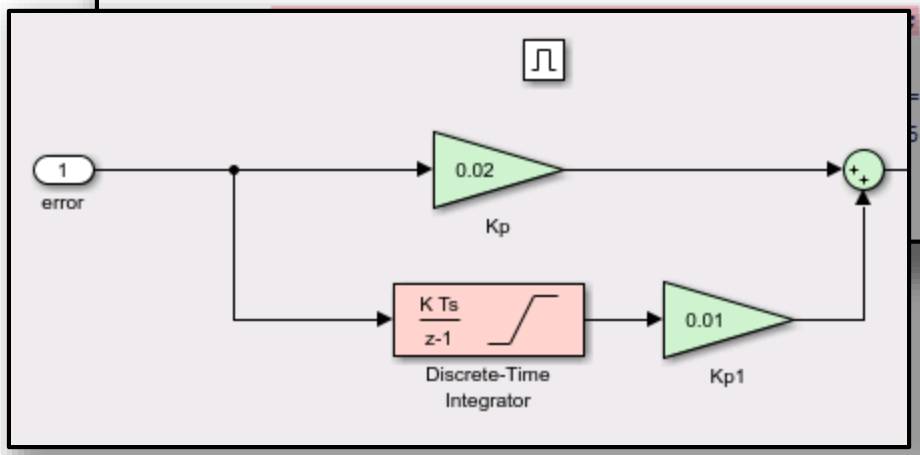
# Coverage for Generated Code (R2016a)




**cruise_control (SIL)**

**Press Play**

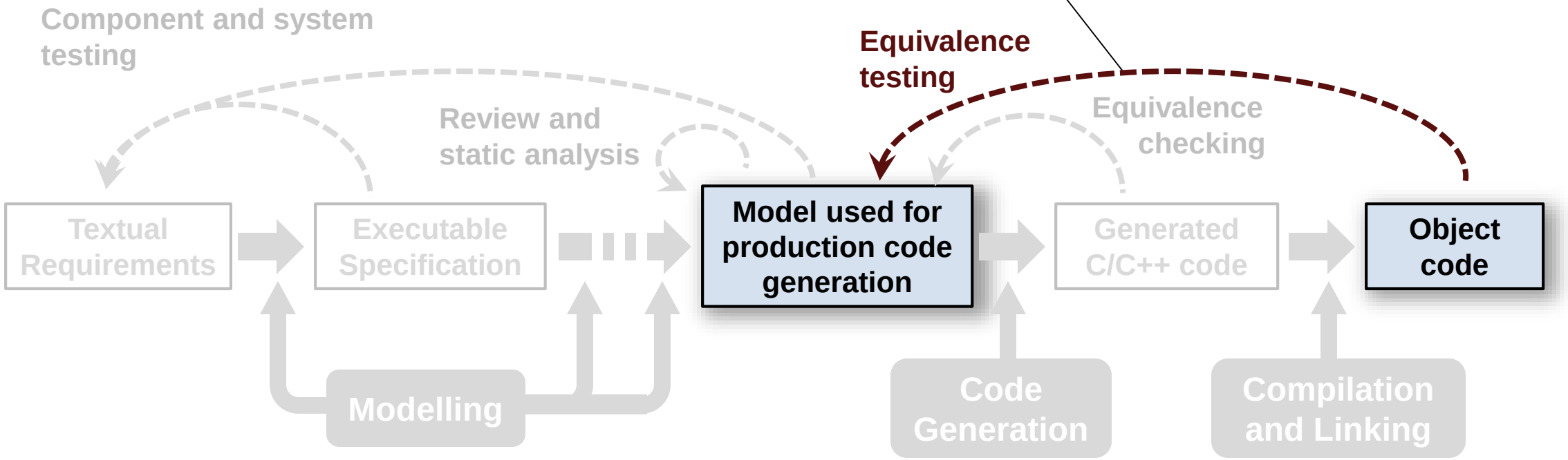
```
77    if (rtb_ActiveControl) {
78      /* Sum: '<S2>/Sum' incorporates:
79       *  DiscreteIntegrator: '<S2>/Discrete-Time Integrator'
80       *  Gain: '<S2>/Kp'
81       *  Gain: '<S2>/Kp1'
82       */
83      *rty_throt = 0.02 * rtb_Switch2 + 0.01 *
84        localDW->DiscreteTimeIntegrator_DSTATE;
85
86      /* Update for DiscreteIntegrator: '<S2>/Discrete-Time Integrator' *
87      localDW->DiscreteTimeIntegrator_DSTATE += 0.01 * rtb_Switch2;
88      if (localDW->DiscreteTimeIntegrator_DSTATE >= 5.0) {
```



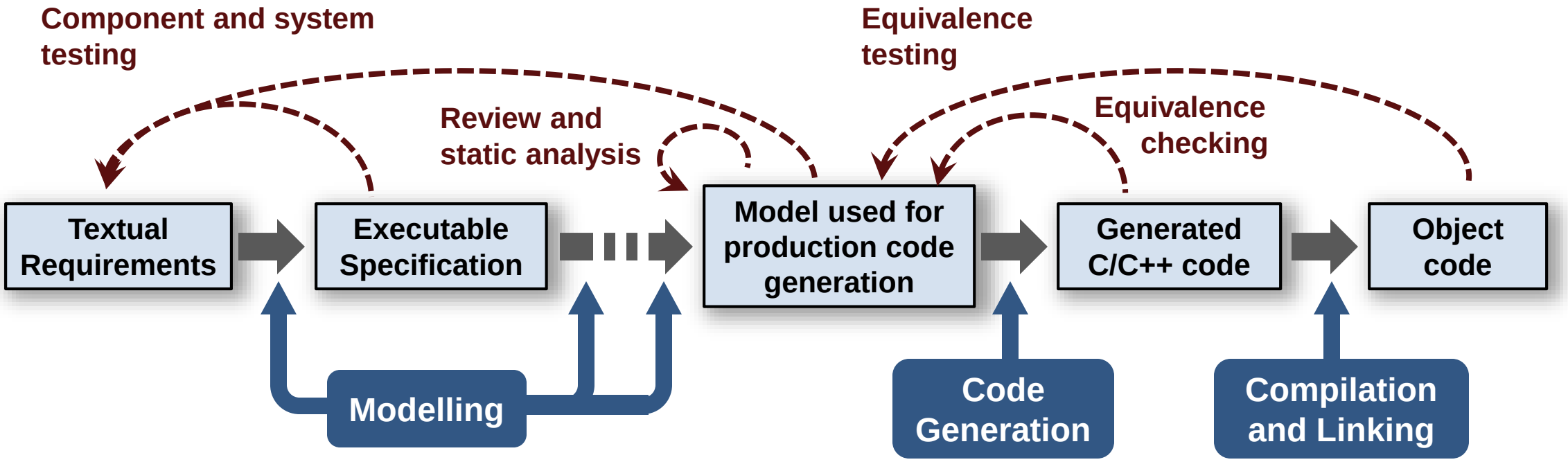

**Can also be highlighted on model**

# Model Based Design Verification Workflow

- Perform PIL Testing
- Perform HIL Testing

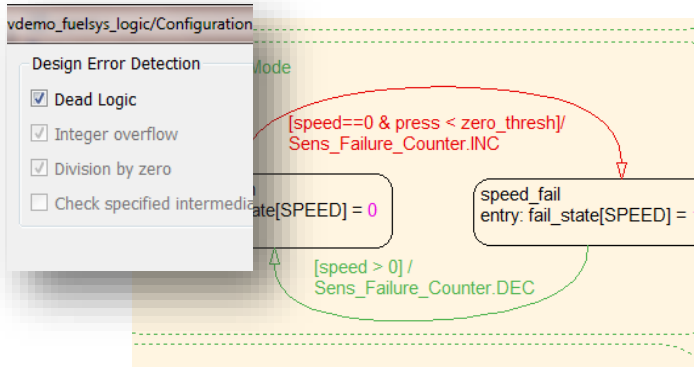


# Model Based Design Verification Workflow

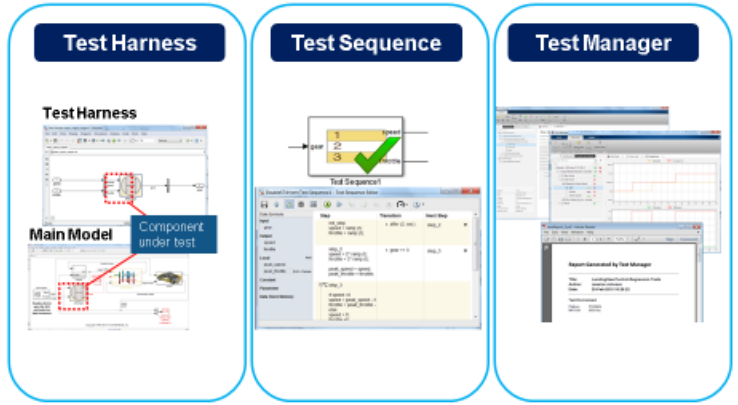


# Systematic Verification

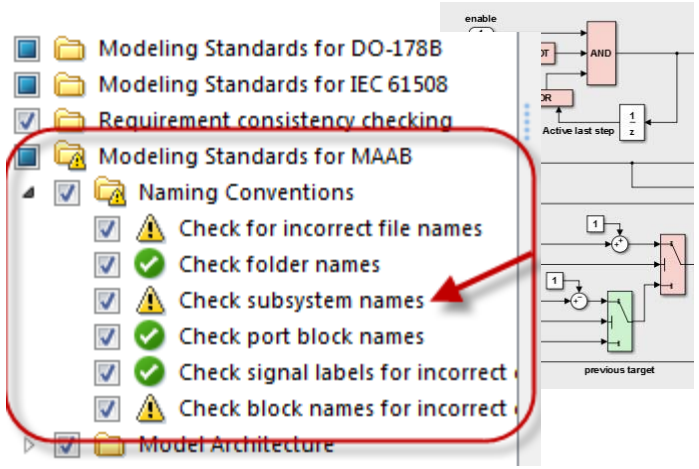
- Ensure that verification is systematically performed across:
  - All requirements
  - Complete model structure
  - Complete code structure
  - All design behaviors




**Simulink Design Verifier**




**Simulink Test**

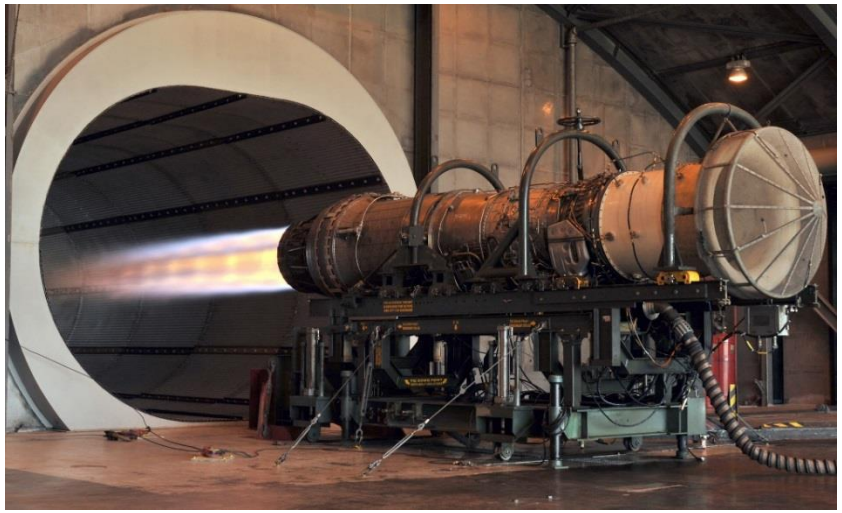



**Simulink Verification & Validation**

# Test and Verification

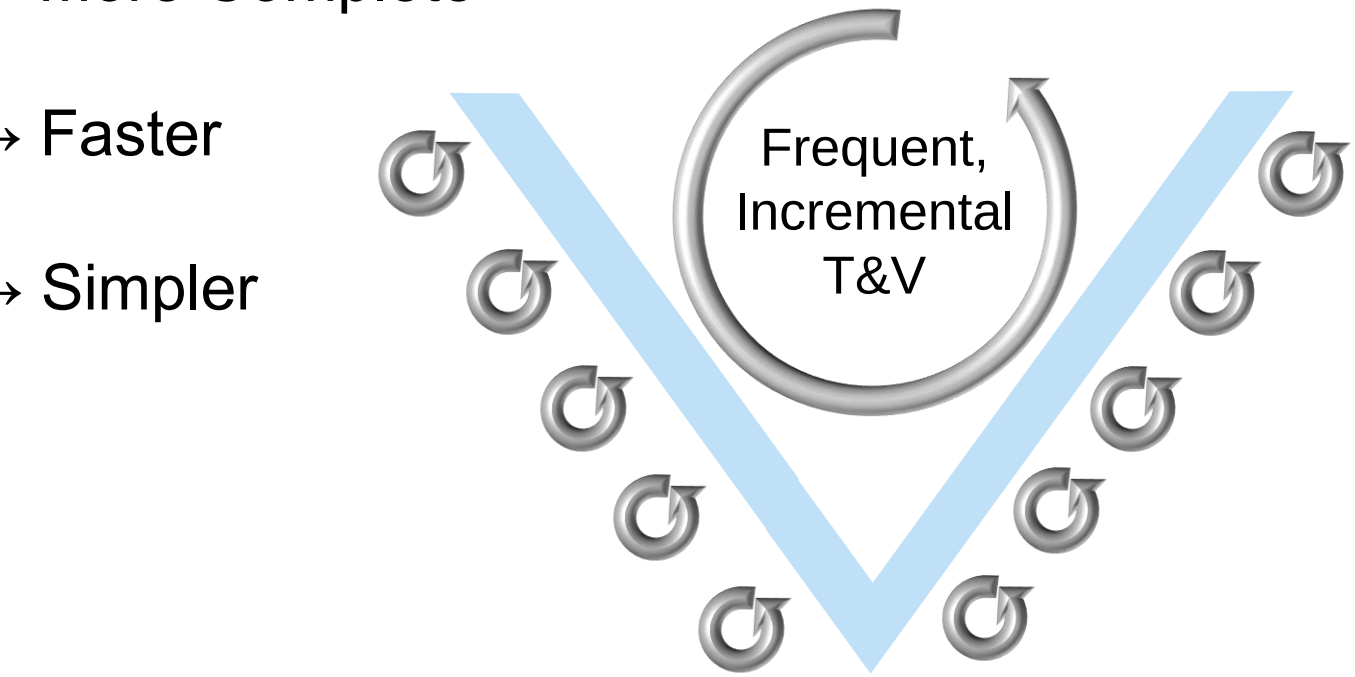
- Essential
- Expensive
- Complex

} Pain Points



# Test and Verification

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



# Thank You!