

# Increasing Design Confidence

*Model and Code Verification*



## The Cost of Failure...

Ariane 5



**\$7,500,000,000**

Rocket & payload lost

## The Cost of Failure...

USS Yorktown



# 0 Knots

Top speed

## The Cost of Failure...

Therac-25



# 6 Casualties

due to radiation overdose

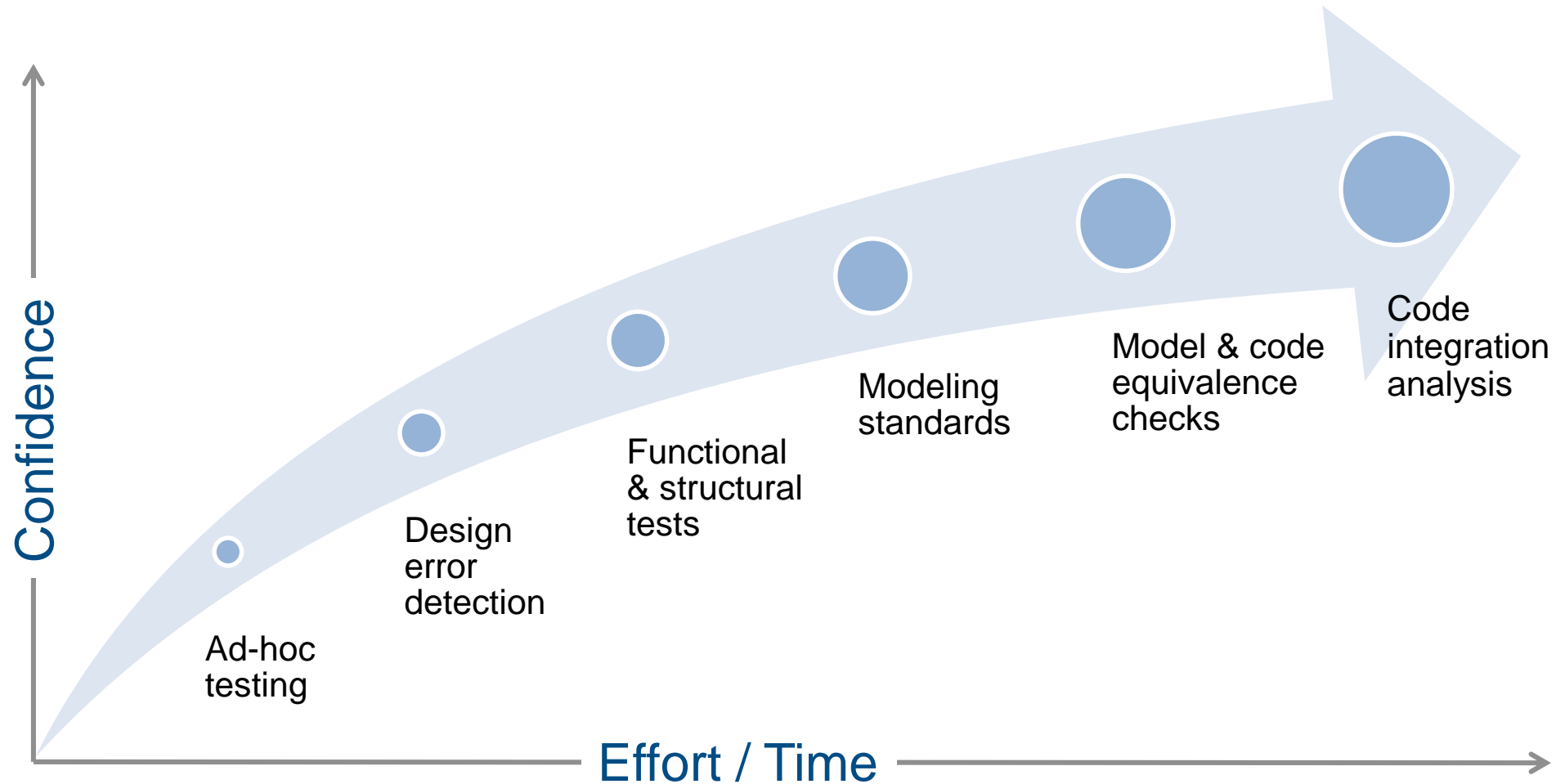
# Motivation

*It is easier and less expensive to fix design errors early in the process when they happen.*

## **Model-Based Design enables:**

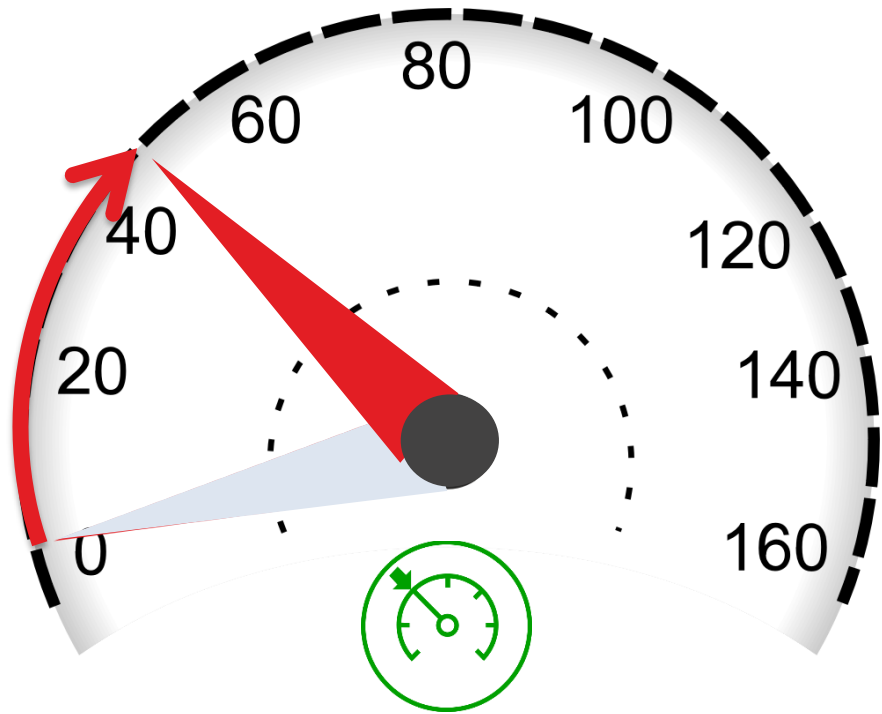
- 1. Early testing to increase confidence in your design*
- 2. Delivery of higher quality software throughout the workflow*

# Gaining Confidence in our Design



# Application: Cruise Control

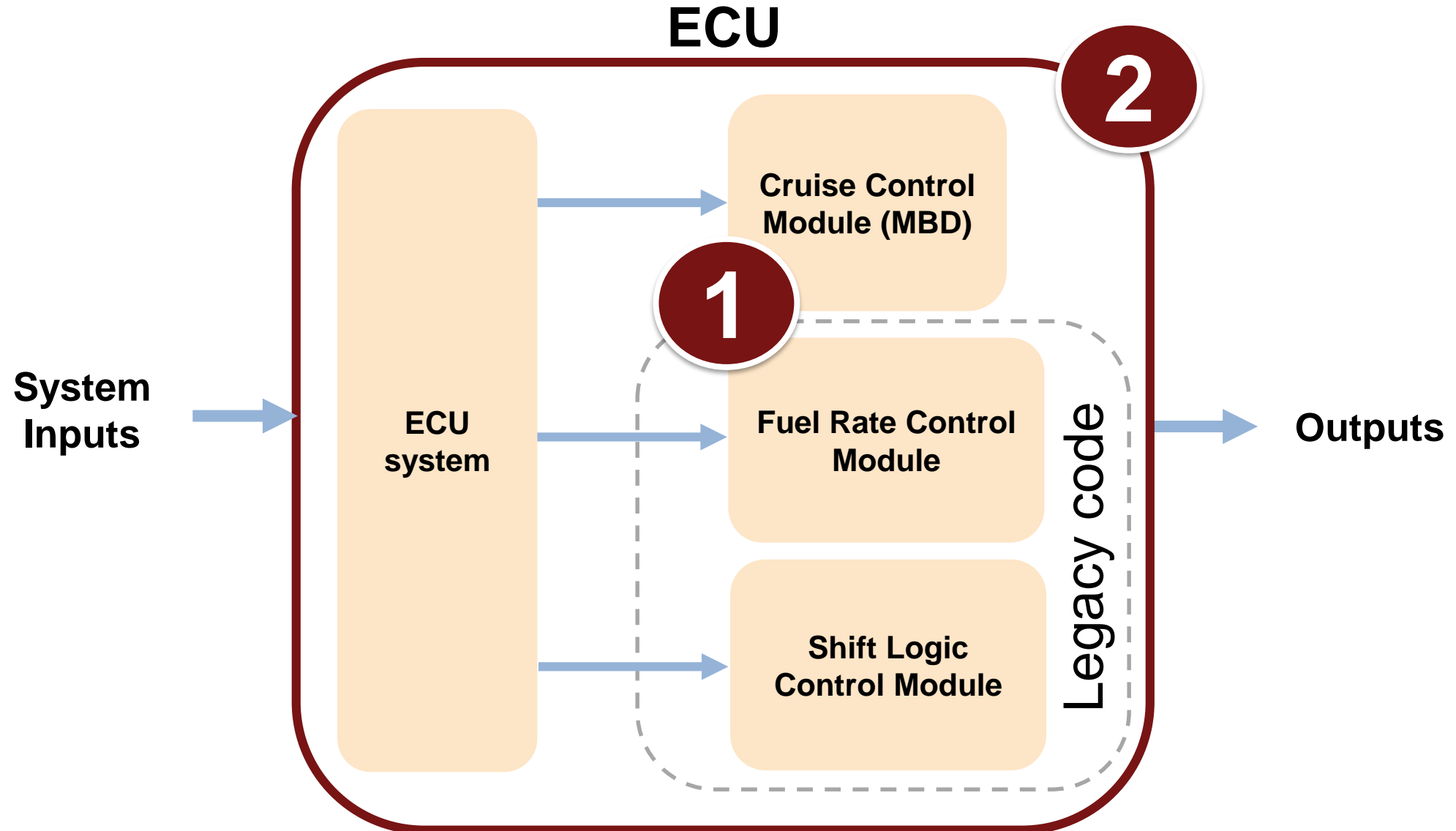
Control speed according to setpoint



50 km/h

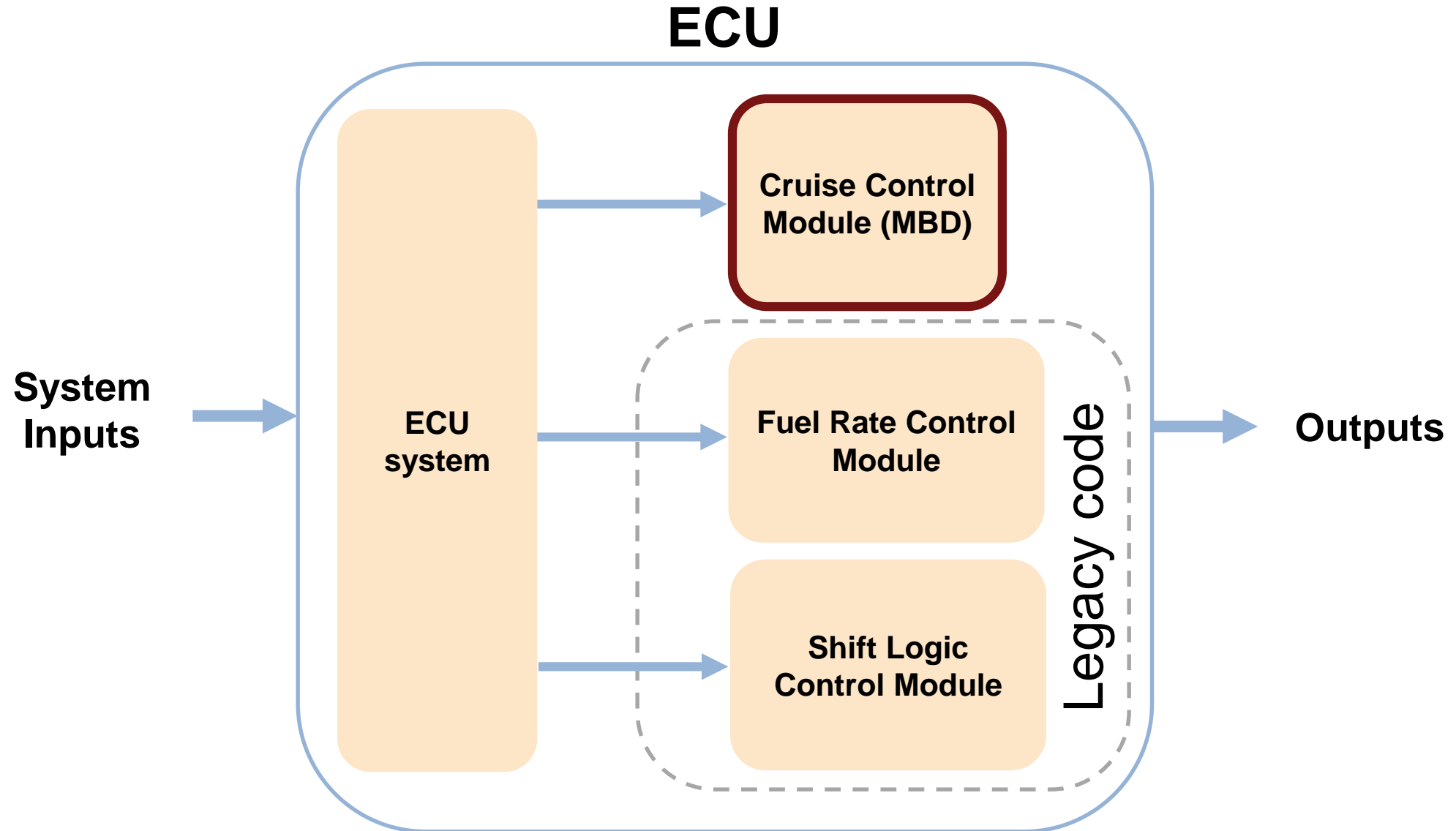


# Application: Cruise Control

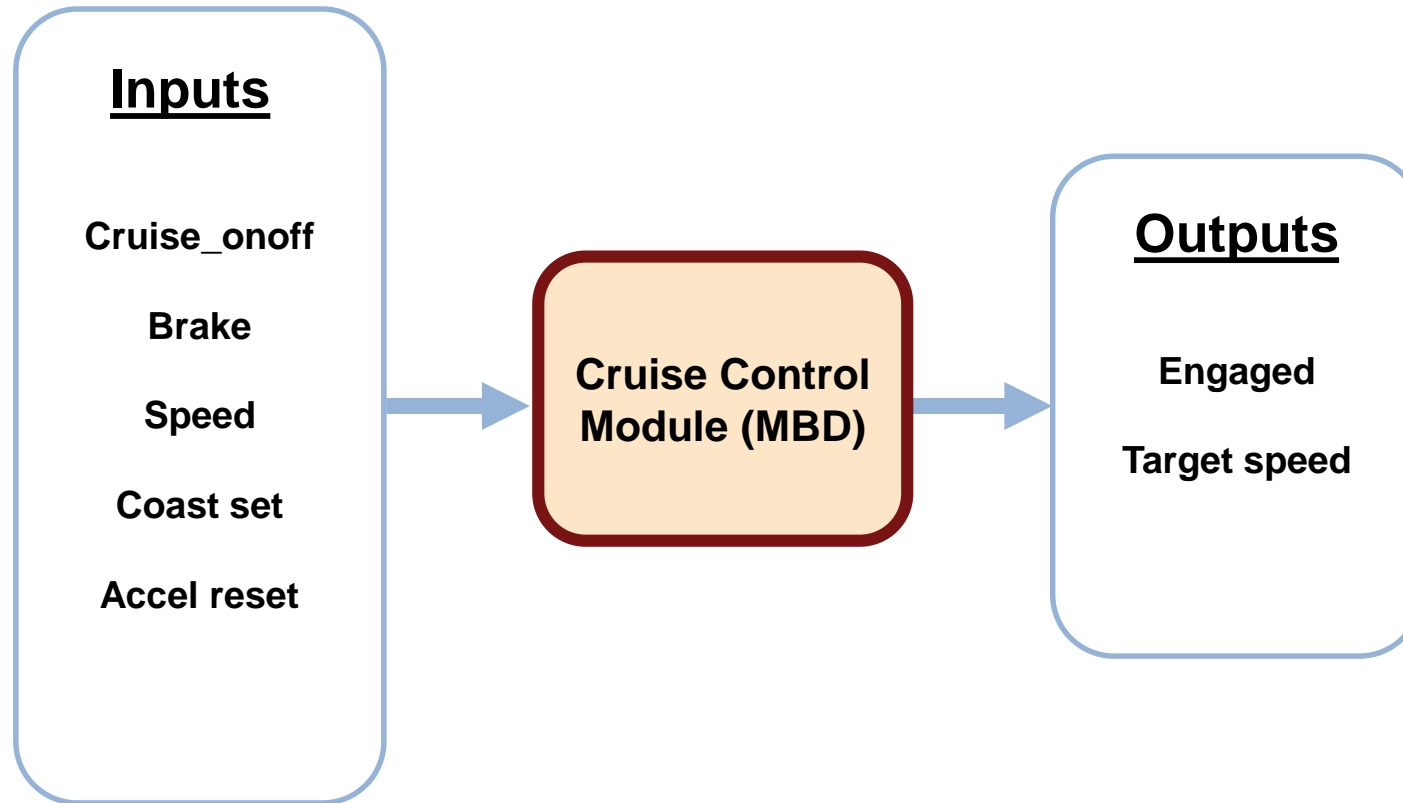




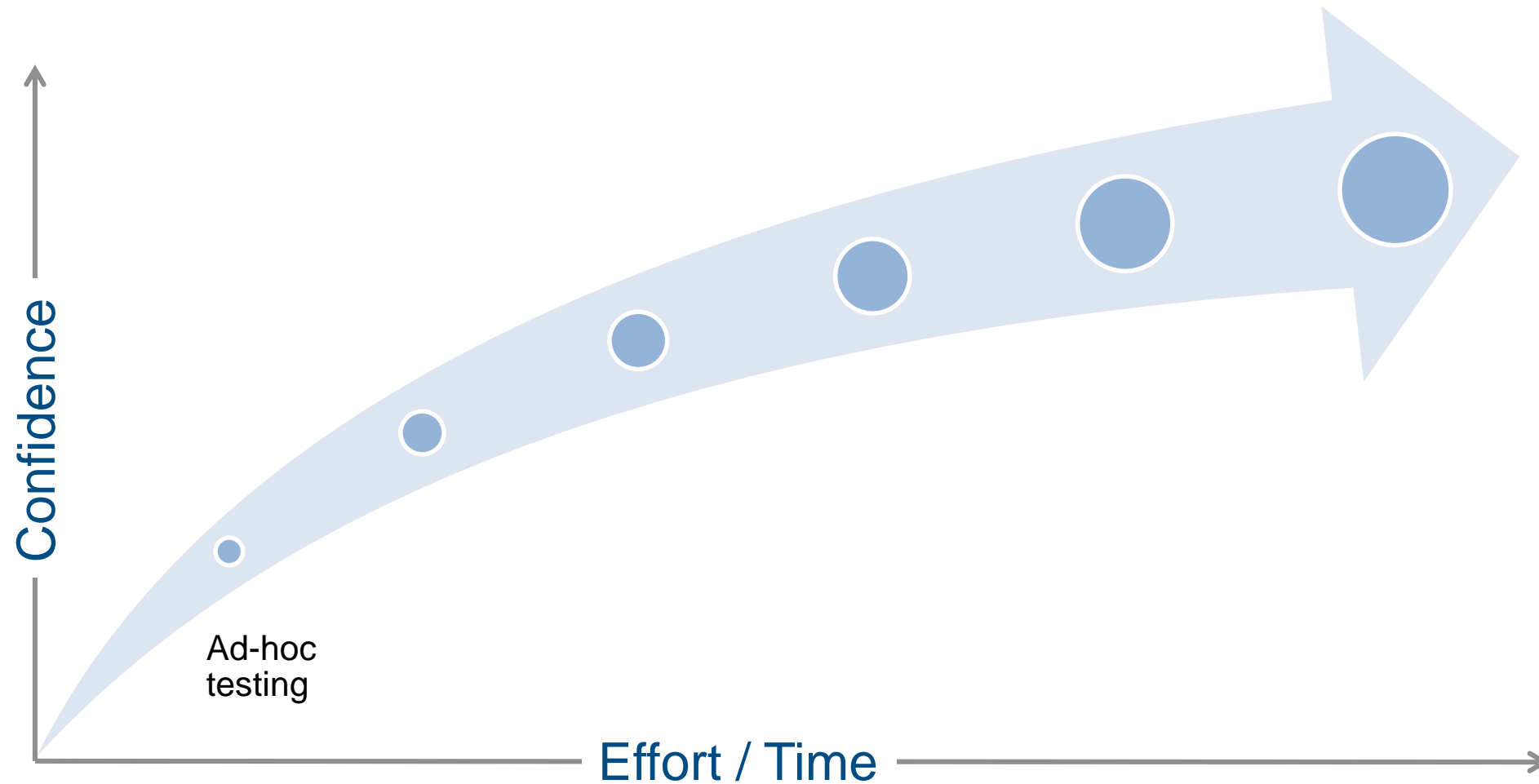
# Application: Cruise Control



# Application: Cruise Control



# Gaining Confidence in our Design



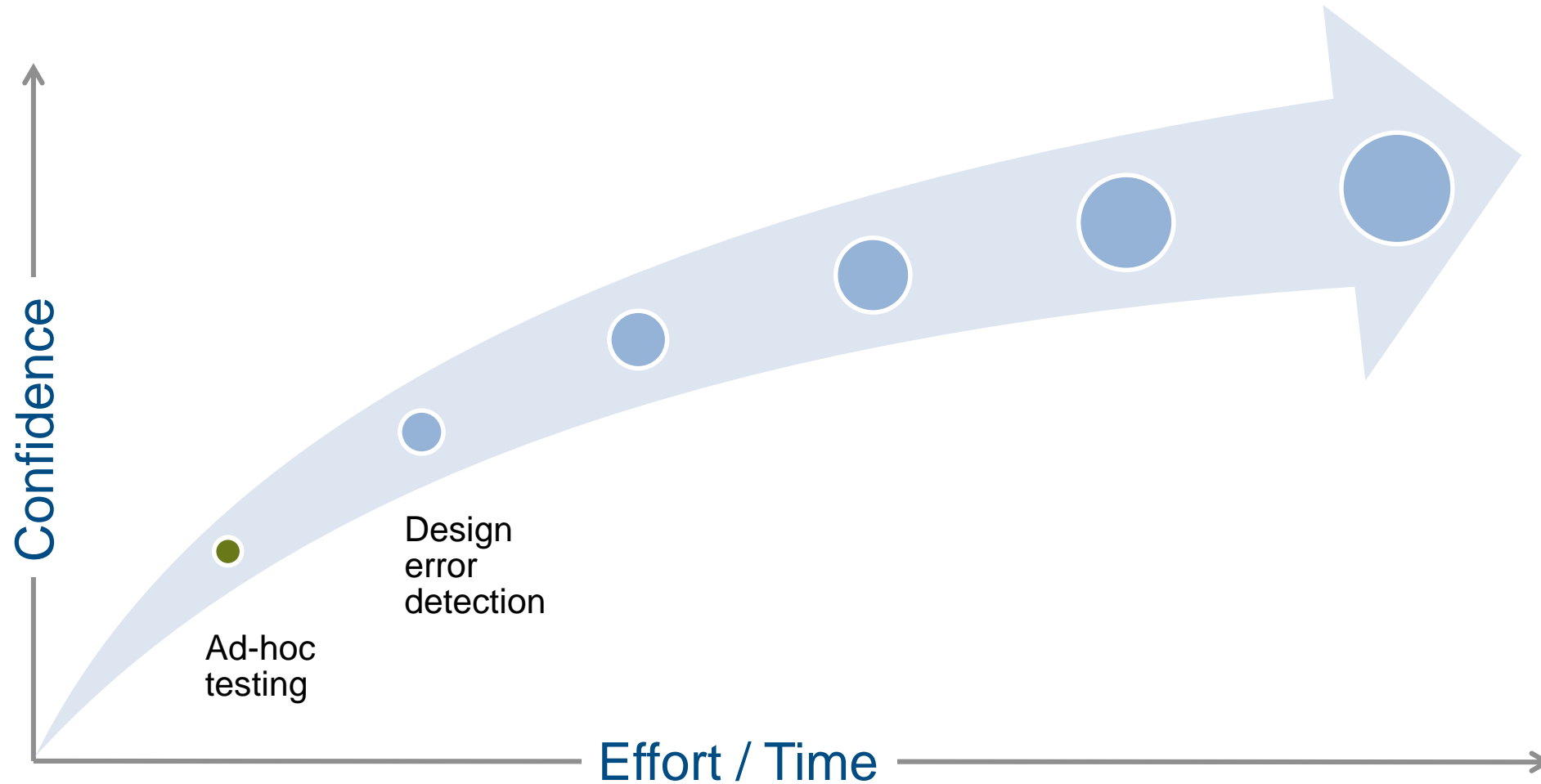
# Ad-hoc Tests

Dashboard blocks facilitate early ad-hoc testing

The image shows two Simulink windows. The left window displays the 'Dashboard' for 'CruiseControl\_RP\_level\_harness', featuring several interactive blocks: three 'On/Off' toggle switches for 'Cruise', 'CoastSet', and 'AccelRes'; a 'VehicleSpeed' gauge with a needle pointing to approximately 26; and another 'On/Off' toggle switch for 'Brake'. The right window shows the 'Test Unit' for 'CruiseControl\_RP\_level', which is a blue block with inputs for 'CruiseOnOff' (boolean), 'Brake' (boolean), 'Speed' (single), 'CoastSetSw' (single), and 'AccelResSw' (single). It has two outputs: 'engaged' (boolean) and 'tspeed' (single). The 'engaged' output is currently set to '1' and 'engaged', while the 'tspeed' output is set to '26' and 'speed'.

The image shows a Stateflow chart titled 'Stateflow (chart) CruiseControl\_RP\_level/Compute target speed - Simulink'. The chart is divided into three main states: 'OFF', 'STANDBY', and 'CRUISE'.  
 - The 'OFF' state has an entry condition 'en: engaged = false; tspeed = 0;'. It transitions to 'STANDBY' on the event '[CruiseOnOff]' and back to 'OFF' on '[~CruiseOnOff]'.  
 - The 'STANDBY' state has an entry condition 'en: engaged=false;'. It transitions to 'CRUISE' on the event '[Brake || ... Speed > maxspeed || ... Speed < minspeed]'.  
 - The 'CRUISE' state is further divided into three sub-states: 'Accel', 'Steady', and 'Coast'.  
 - The 'Accel' sub-state has an entry condition 'en: engaged = true;'. Its action is 'tspeed = tspeed + inodec;'. It transitions to 'Steady' on the event '[hasChangedTo(AccelResSw,true) ... && tspeed < maxspeed]'.  
 - The 'Steady' sub-state has an action '(tspeed = Speed;)' and transitions to 'Accel' on '[AccelResSw || ... tspeed >= maxspeed]' and to 'Coast' on '[CoastSetSw || ... tspeed <= minspeed]'.  
 - The 'Coast' sub-state has an action 'tspeed = tspeed - inodec;'. It transitions to 'Steady' on '[CoastSetSw,true] ... && tspeed > minspeed]' and back to 'Accel' on '[hasChangedTo... (CoastSetSw,true)]'.  
 - There are also transitions between 'Accel' and 'Coast' sub-states based on 'CoastSetSw' events.

# Gaining Confidence in our Design



# Finding Design Errors: Dead Logic

The image displays the Simulink Design Verifier interface and a Stateflow chart for a cruise control system. The Design Verifier window shows the following configuration and results:

**Configuration Parameters: CruiseControl\_IntCalc/ModelReferencin...**

Select:

- Solver
- Data Import/Export
- Optimization
- Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
- Code Generation
- Design Verifier
  - Block Replacements
  - Para
  - Test
  - Desi
  - Prop
  - Resu
  - Repd

**Design Error Detection**

- Dead logic
- Identify active logic
- Integer overflow
- Division by zero
- Check specified intermediate minimum an
- Out of bound array access

**Simulink Design Verifier ...**

Close results

**Design error detection completed normally.**

2/70 objectives are dead logic.

68/70 objectives are active logic.

**Results:**

- [Generate detailed analysis report](#)
- [Open harness model](#)

**Stateflow (chart) CruiseControl\_IntCalc/Compute target speed \* - Simulink**

File Edit View Display Chart Simulation Analysis Code Tools Help

CruiseControl\_IntCalc Compute target speed

Compute target speed.CRUISE.ON."[after(incdec/holdrate... \*..."

Transition: Transition trigger expression F **DEAD LOGIC**

Transition: Transition trigger expression T **ACTIVE LOGIC**

[after(incdec/holdrate... \*10,tick)]

Coast

tspeed = tspeed - incdec

[after(incdec/holdrate... \*10,tick)]

View 4 warnings 81% FixedStepDiscrete

# Finding Unintended Behavior

Command Window

```
debug>> incdec
incdec =
     1

debug>> holdrate
holdrate =
     5
```

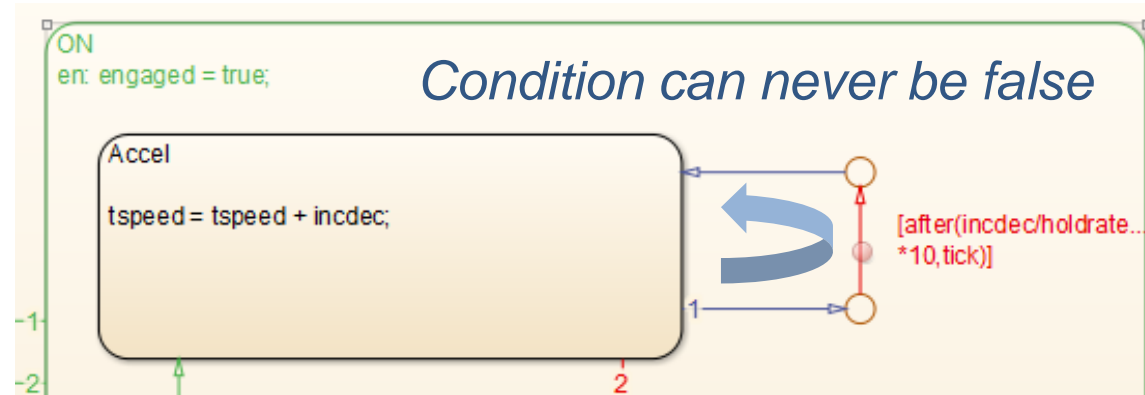
```
debug>> class(incdec)
ans =
uint8

debug>> class(holdrate)
ans =
uint8
```

```
debug>> incdec/holdrate*10
ans =
     0
```

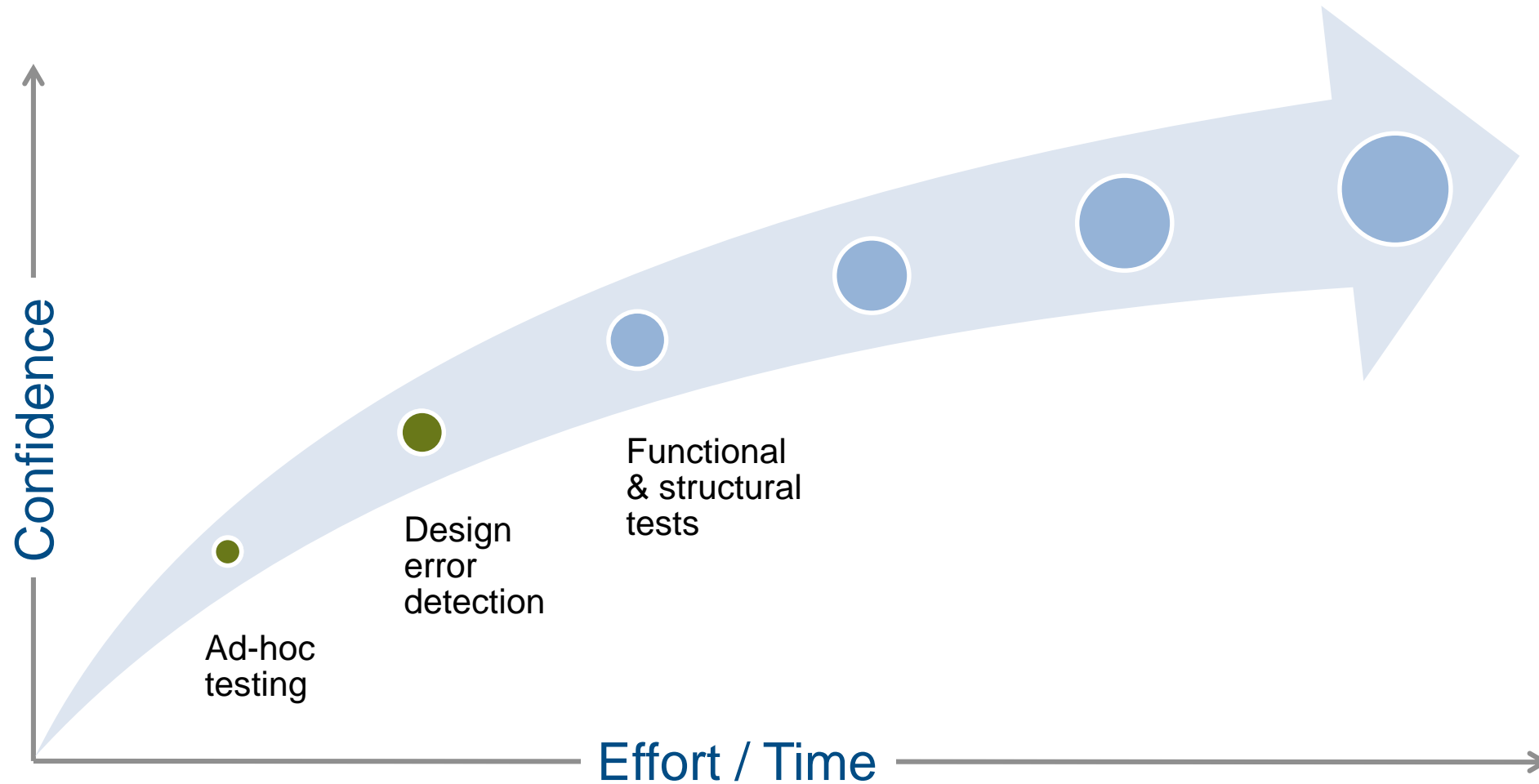
```
debug>> 10*incdec/holdrate
ans =
     2
```

fx debug>>



- **Dead logic** due to “uint8” operation on **incdec/holdrate\*10**
- **Fix** change the order of operation **10\*incdec/holdrate**

# Gaining Confidence in our Design





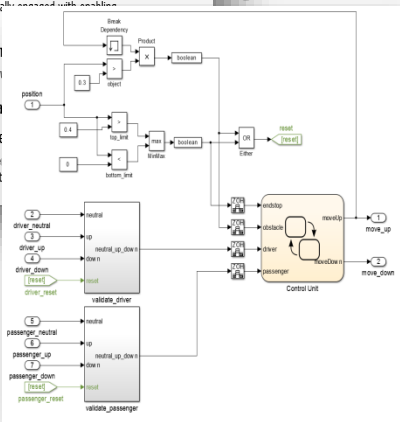
# Simulation Testing Workflow

## Requirements

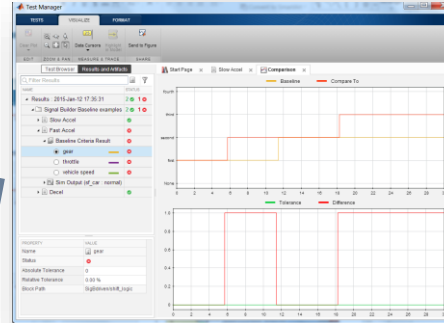
### 2. Functional Requirements

- 2.1.1. **Disabled (off) during start-up and not engaged (inactive)**  
Initial state of cruise control system shall be disabled.
- 2.1.2. **Not engaged (inactive) with enabling (on)**  
The cruise control system shall not be initiated without the vehicle speed.
- 2.1.3. **Disengaged (not active) when**  
The cruise control system shall disengage when the driver or passenger intervenes.
- 2.1.4. **Initial transition from disengaged only with "Set Speed" input after**  
The cruise control shall only transition to engaged when enabled with a "Set Speed" input speed.

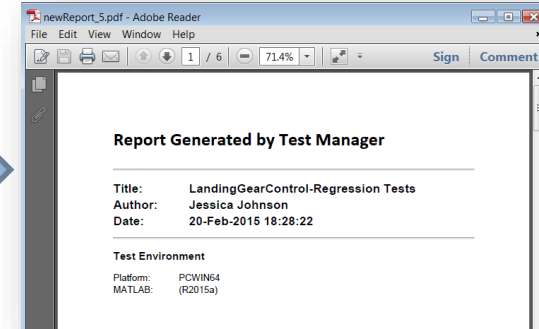
## Design



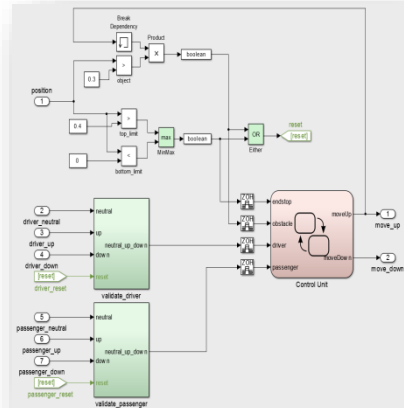
Did we meet requirements?



Review functional behavior



Did we completely test our model?



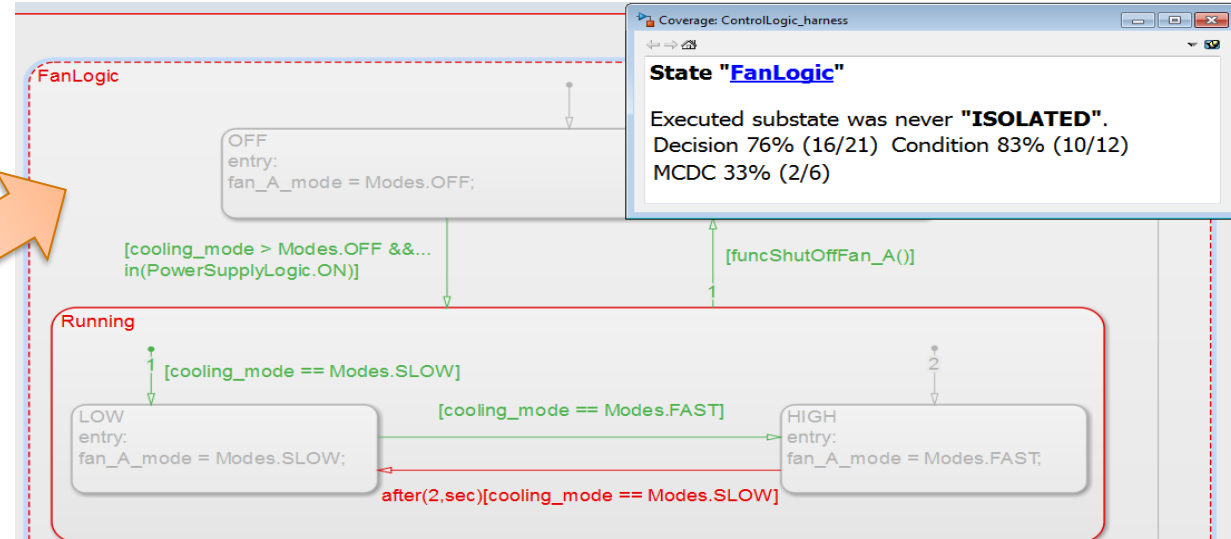
Structural coverage report

	D1	C1	MCDC
1. <a href="#">slvvdemo_powerwindow_controller</a>	58 100%	80%	60%
2. <a href="#">control</a>	57 100%	80%	60%
3. <a href="#">SF: control</a>	56 100%	80%	60%
4. <a href="#">SF: safe</a>	52 100%	75%	50%
5. <a href="#">SF: driverDown</a>	7 100%	75%	50%
6. <a href="#">SF: driverNeutral</a>	26 100%	75%	50%
7. <a href="#">SF: passengerDown</a>	7 100%	75%	50%
8. <a href="#">SF: passengerUp</a>	7 100%	75%	50%
9. <a href="#">SF: driverUp</a>	7 100%	75%	50%

Functional  
Structural

# Did We Completely Test our Model?

## Model Coverage Analysis



Potential causes of less than 100% coverage:

- Missing requirements
- Over-specified design
- Design errors
- Missing tests

## Summary

Model Hierarchy/Complexity:		Test 1					
		D1	C1	MCDC			
1. <a href="#">ControlLogic_modelingcompleted</a>	101	84%	83%	53%			
2. . . . <a href="#">ModeLogic</a>	100	84%	83%	53%			
3. . . . . <a href="#">SF: ModeLogic</a>	99	84%	83%	53%			
4. . . . . . <a href="#">SF: CoolingSystemLogic</a>	70	81%	87%	58%			
5. . . . . . . <a href="#">SF: CoolingSystemA</a>	35	77%	82%	50%			
6. . . . . . . . <a href="#">SF: FanLogic</a>	14	76%	83%	33%			
7. . . . . . . . . <a href="#">SF: Running</a>	6	90%	75%	0%			
8. . . . . . . . . .	100%						

# Requirements Based Functional Testing with Coverage Analysis

- All 14 requirements based test cases pass
- By analyzing model coverage results we find:
  - Missing test cases for vehicle speed exit conditions, and
  - Missing requirements (and test cases) for “hold” or continuous speed button input

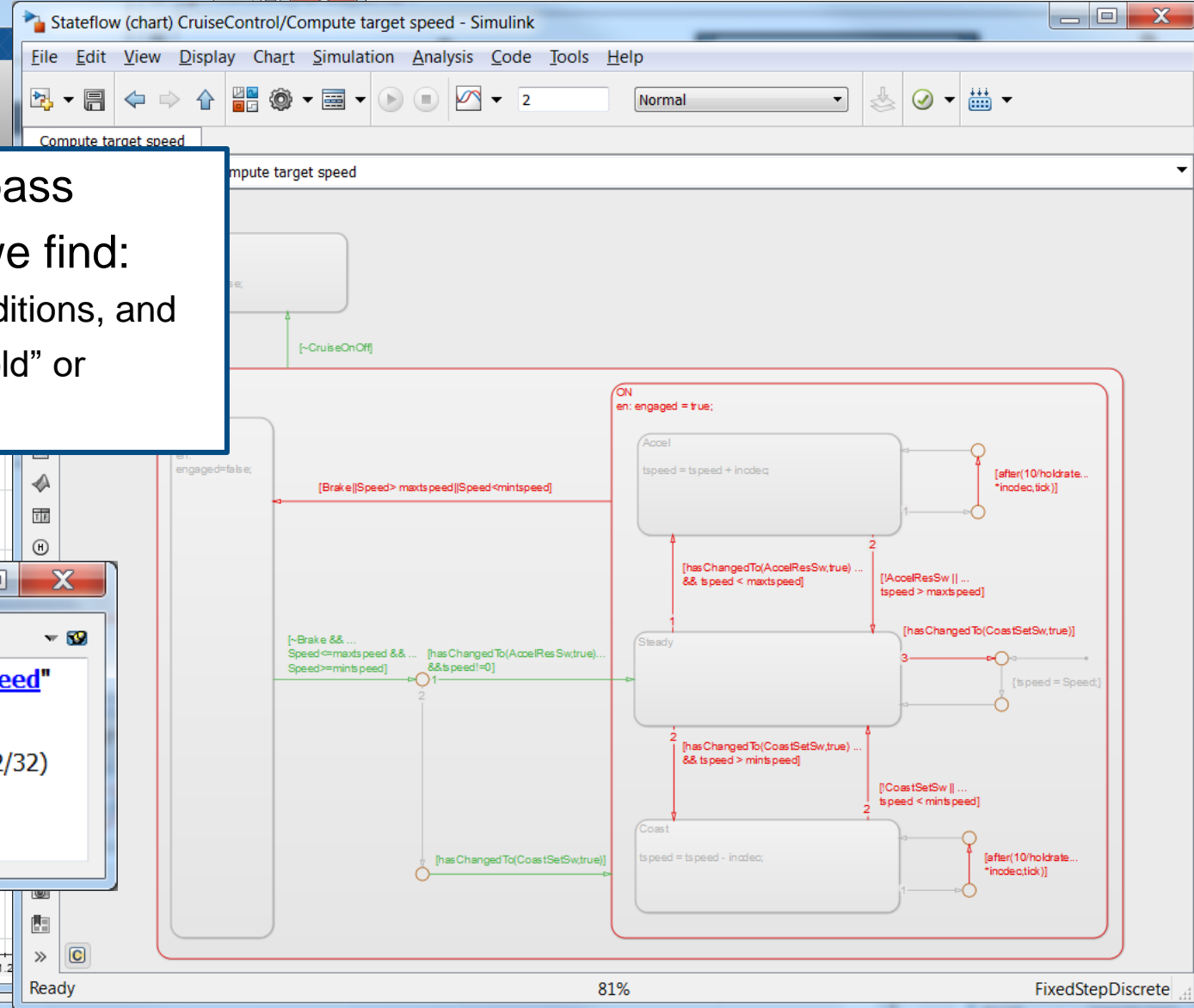
Coverage: CruiseControl\_Harness\_...

SubSystem block "Compute target speed"

Decision 82% (31/38) Condition 69% (22/32)  
 MCDC 50% (8/16)

PROPERTY	VALUE
Name	tspeed
Status	✓
Absolute Tolerance	0
Relative Tolerance	0%
Block Path	CruiseControl_Harness...

PROPERTY	VALUE
Name	tspeed
Status	✓
Absolute Tolerance	0
Relative Tolerance	0%
Block Path	CruiseControl_Harness...



# Functional Testing with Added Requirements & Test Cases

The image displays the MATLAB Test Manager interface. On the left, the 'TESTS' tab shows a list of test cases under the folder '02\_Functional\_Baseline\_Excel\_Fu'. The 'Results and Artifacts' section shows a table of test results:

NAME	STATUS
02_Functional_Baseline_Excel_Fu	19 ✓
Init_Not_Enable_or_Engaged	✓
Not_Engaged_with_Enable	✓
Disengage_with_Disable	✓
Engage_1st_with_CoastSetSW	✓
No_Engage_1st_with_AccelRes	✓
Disengage_with_Brake	✓
Re_Eng_with_CoastSetSw	✓
Re_Eng_with_AccelResSw	✓
Engd_Decel_w_CoastSetSW	✓
Engd_Accel_w_AccelResSW	✓
Engd_Decel_w_CstSet_Hld	✓

The main window shows the Stateflow chart for 'Compute target speed'. The chart has states: OFF, CRUISE, STANDBY, ON, Steady, and Coast. Annotations on the right side of the chart include:

- 2  $[!AccelResSw \parallel \dots \underline{tspeed} \geq \text{maxtspeed}]$
- 3  $[hasChangedTo(CoastSetSw, true)]$
- 2  $[!CoastSetSw \parallel \dots \underline{tspeed} \leq \text{mintspeed}]$

A 'Coverage: CruiseControl\_Harness...' window is overlaid on the chart, showing:

Chart "Compute target speed"  
Full Coverage

At the bottom of the Stateflow window, the status is 'Ready', '80%', and 'FixedStepDiscrete'.

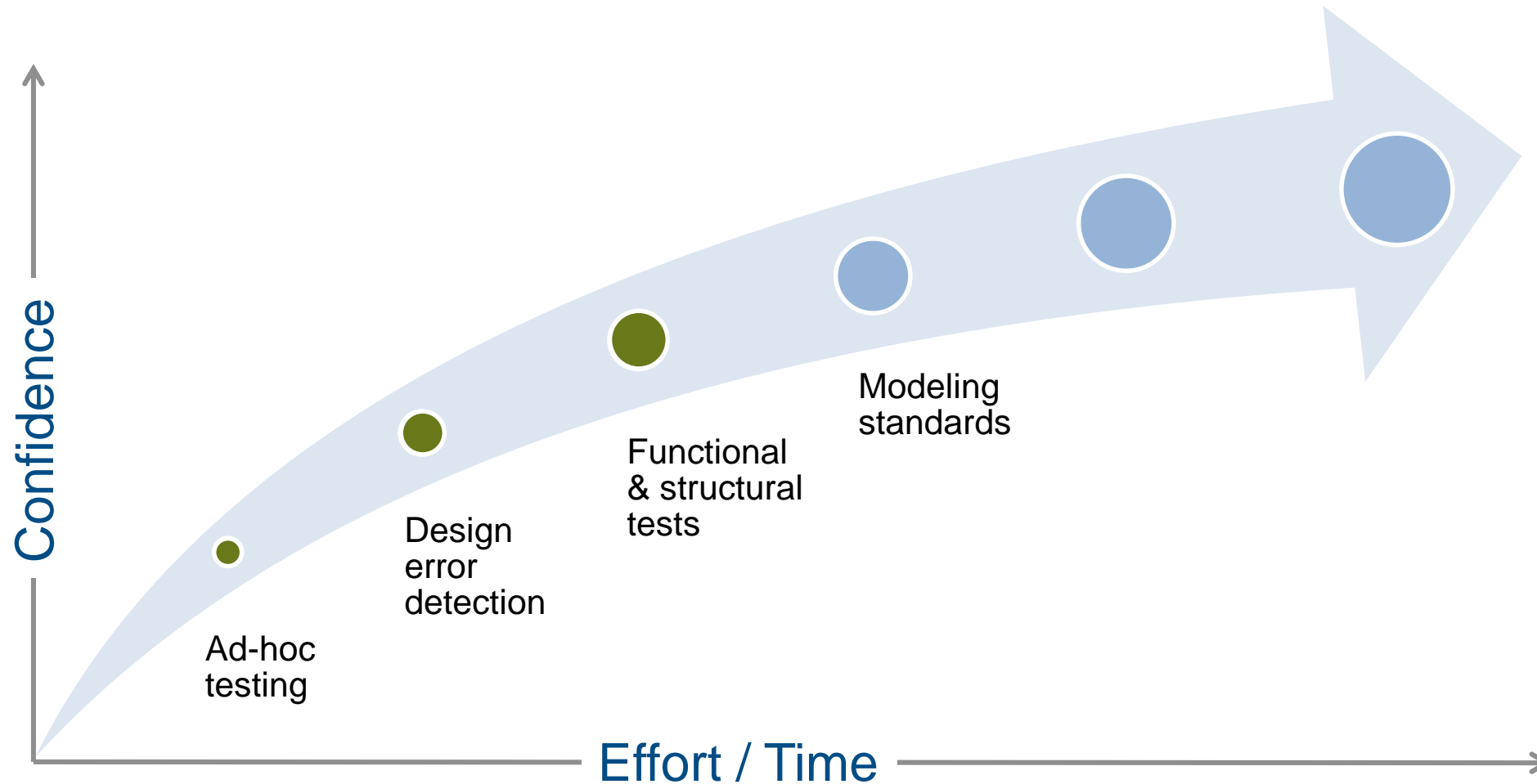
# Functional Testing with Added Requirements & Test Cases

The screenshot displays the MathWorks Test Manager interface. The 'TESTS' tab is active, showing a list of test cases under the '02\_Function' folder. A large white box in the center contains the following text:

- Added 2 new requirements for the “hold” case for speed setting input buttons
- Added 5 test cases to the original 14 requirements based test cases
  - 3 test cases for the 2 new requirements
  - 2 test cases for the missing test cases for the vehicle speed exist conditions
- 4/5 new functional test cases pass
  - Failed test case showed overshoot beyond target speed limits
  - Coverage analysis highlighted transitions with design errors
  - Fixed comparison operators, ( $<$ )  $\rightarrow$  ( $\leq$ ), and ( $>$ )  $\rightarrow$  ( $\geq$ )
- Now all (19) functional test cases pass with 100% model coverage!**

The background shows the Stateflow chart for 'CruiseControl/Compute target speed - Simulink'. The chart includes transitions with guards such as `[!CoastSetSw || ... tspeed <= mintspeed]` and `[!CoastSetSw || ... tspeed <= mintspeed]`. A 'Full Coverage' status is visible in the bottom left corner of the chart window.

# Gaining Confidence in our Design



# Model Advisor – Model Standards Checking

**Model Advisor - Step\_02\_fuelsys**

File Edit Run Settings Highlighting Help

Find:

**Modeling Standards for MAAB**

Model Advisor

Analysis

Group of MAAB checks

Run Selected Checks 3.

Show report after run 2.

Report

Report: [...report\\_533.html](#) Save As...

Date/Time: Not Applicable

Summary:  Pass: 0  Fail: 0  Warning: 0

Tips

Legend

^ Running this check triggers an Update Diagram.

1.  Modeling Standards for MAAB

Upgrade Advisor

Code Generation Advisor

Performance Advisor

Web Browser - Model Advisor Report for 'Step\_02\_fuelsys'

Model Advisor Report for 'Step\_02\_fuelsys' x +

Location: file:///C:/fhabring/Projects/CRE/vwtWorkshop/Work/slprj/modeladvisor/Step\_02\_fuelsys/report\_531

**Model Advisor Report - Step\_02\_fuelsys.slx**

Simulink version: 8.2  
System: Step\_02\_fuelsys

Model version: 1.242  
Current run: 20-Sep-2013 08:45:50

**Run Summary**

<input checked="" type="checkbox"/> Pass	<input checked="" type="checkbox"/> Fail	<input checked="" type="checkbox"/> Warning	<input checked="" type="checkbox"/> Not Run	Total
41	0	11	0	52

**Modeling Standards for MAAB**

**Naming Conventions**

**Check file names**

Identify file names with incorrect characters or formatting.

**See Also**

- [MathWorks Automotive Advisory Board Guideline: ar\\_0001](#)

**Passed**

All files have correct names.

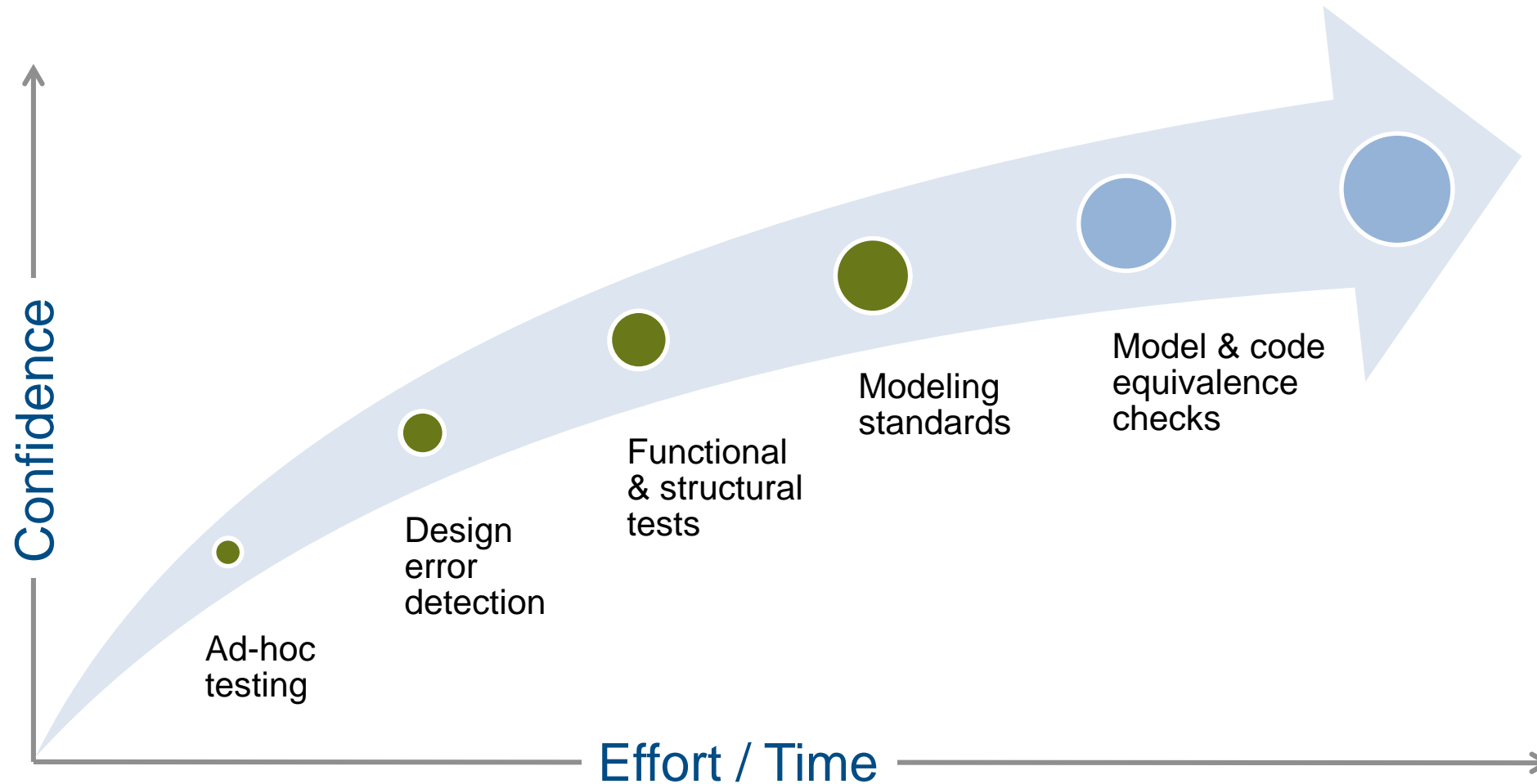
**Check folder names**

Identify folders using incorrect characters and formatting.

**See Also**

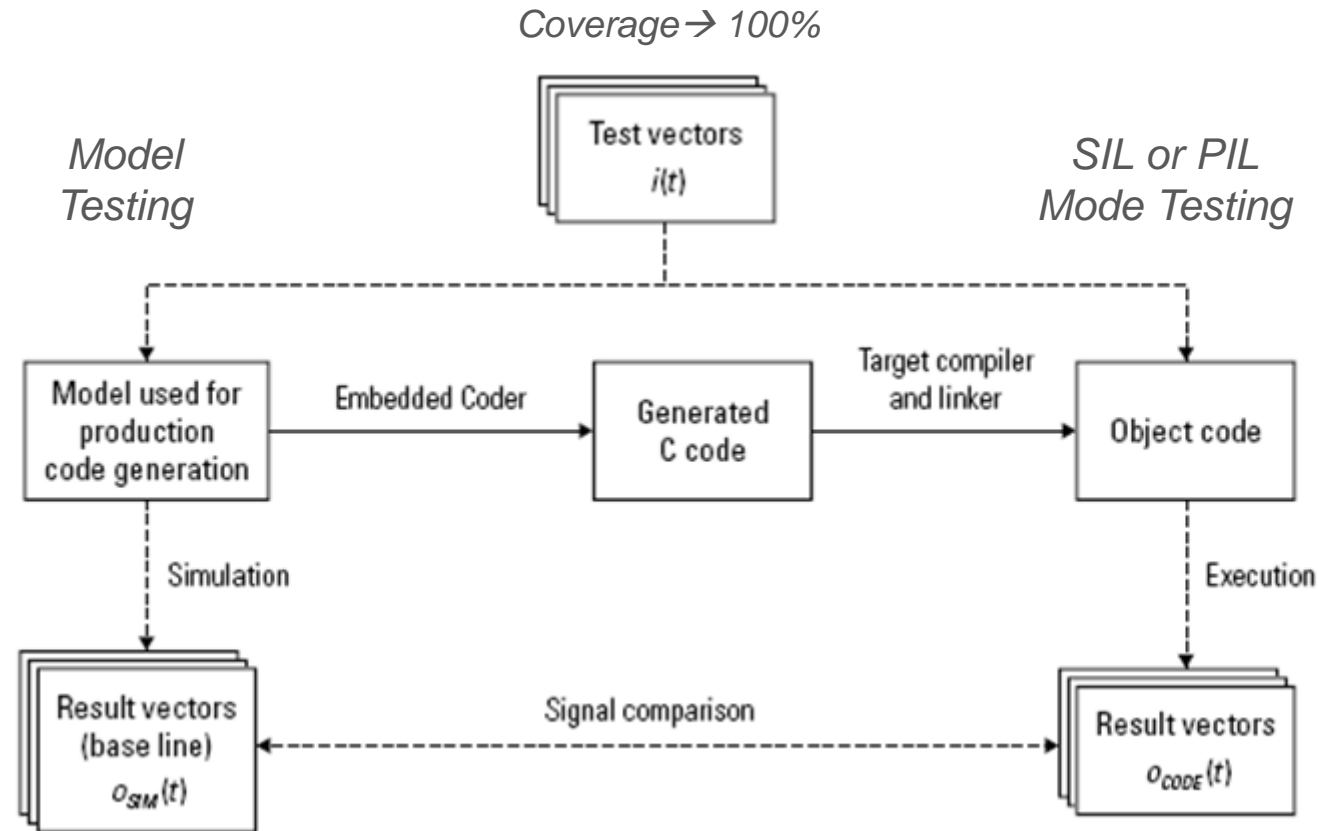
- [MathWorks Automotive Advisory Board Guideline: ar\\_0002](#)

# Gaining Confidence in our Design





# Equivalence Testing: Model vs SIL or PIL Mode Testing



# Code Generation with Model-to-Code Traceability

The image displays the MATLAB/Simulink environment with a Simulink model and a Code Generation Report window.

**Simulink Model:** The model, titled "Step\_07\_logic", shows a "Fault Detection Logic" block. It has four input ports: "1 throttle", "2 speed", "3 EGO", and "4 MAP". The block is connected to a "safety\_logic" block. The status bar indicates "Ready" and "100%" zoom.

**Code Generation Report:** The report window shows the generated code for the "Step\_07\_logic" block. The "Contents" section includes links for "Summary", "Subsystem Report", "Traceability Report", "Static Code Metrics Report", and "Code Replacements Report". The "Generated Code" section lists files: "Model files" (Step\_07\_logic.c, Step\_07\_logic.h, Step\_07\_logic\_private.h, Step\_07\_logic\_types.h), "Shared Utility files" (model\_reference\_types.h, rtw\_shared\_utils.h, rtwtypes.h), and "Interface files" (Step\_07\_logic\_ssf.c, rtiostream.h, rtiostream\_tcpip.c).

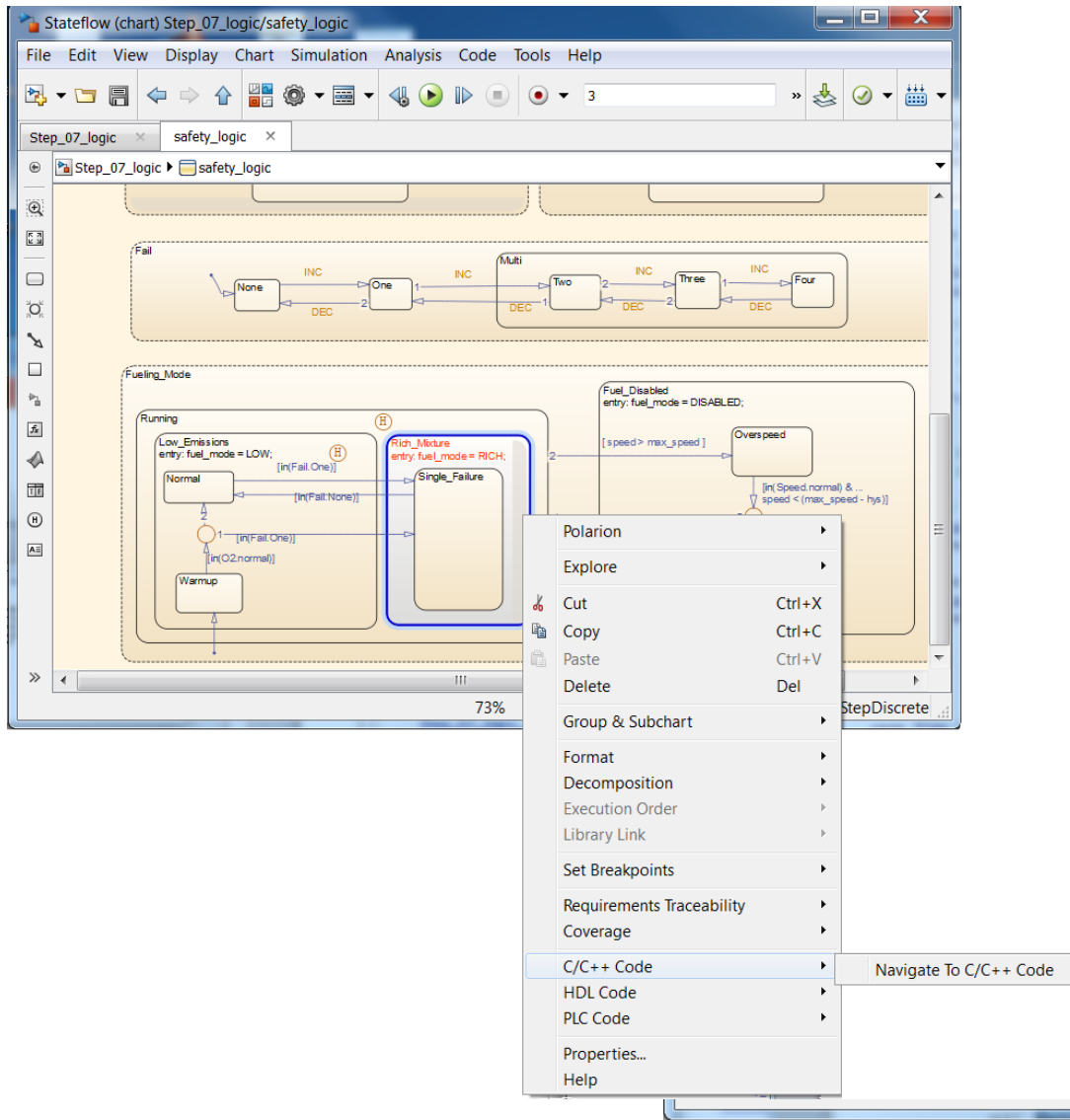
The code snippet shows a switch statement for "Step\_07\_logic\_IN\_Rich\_Mixture". A specific entry is highlighted with a blue box:

```

112 /* Entry 'Rich_Mixture': '<S1>:26'
113 /* Requirements for Entry 'Rich_Mixture': '<S1>:26':
114 * 1. Enriched mixture usage
115 */
116 *rty_fuel_mode = RICH;
117 }
118
119 /* Entry Internal 'Rich_Mixture': '<S1>:26'
120 * Requirements for Entry Internal 'Rich_Mixture': '<S1>:26':
121 * 1. Enriched mixture usage
122 */
123 localDW->is_Rich_Mixture = Step_07_logic_IN_Single_Failure;
124 break;
125
126 default:
127 localDW->is_Running = Step_07_logic_IN_NO_ACTIVE_CHILD;
128 break;
129 }
130 }
131 }
132 break;
133
134 case Step_07_logic_IN_Shutdown:
135 /* During 'Shutdown': '<S1>:29' */
136 if (localDW->sfEvent == Step_07_logic_exit_from_Multi) {
137 /* Transition: '<S1>:63' */
138 localDW->is_Fuel_Disabled = Step_07_logic_IN_NO_ACTIVE_CHILD;

```

# Code Generation with Model-to-Code Traceability



```

95
Sw 96     case Step_07_logic_IN_Warmup:
97         localDW->is_Low_Emissions = Step_07_logic_IN_Warmup;
98         localDW->was_Low_Emissions = Step_07_logic_IN_Warmup;
99         break;
100
=> 101     default:
102         localDW->is_Low_Emissions = Step_07_logic_IN_NO_ACTIVE_CHILD;
103         break;
104     }
105     break;
106
Sw 107     case Step_07_logic_IN_Rich_Mixture:
=>T 108     if (localDW->is_Running != Step_07_logic_IN_Rich_Mixture) {
109         localDW->is_Running = Step_07_logic_IN_Rich_Mixture;
110         localDW->was_Running = Step_07_logic_IN_Rich_Mixture;
111
112         /* Entry 'Rich_Mixture': '<S1>:26'
113          * Requirements for Entry 'Rich_Mixture': '<S1>:26':
114          * 1. Enriched mixture usage
115          */
116         *rty_fuel_mode = RICH;
117     }
118
119     /* Entry Internal 'Rich_Mixture': '<S1>:26'
120     * Requirements for Entry Internal 'Rich_Mixture': '<S1>:26':
121     * 1. Enriched mixture usage
122     */
123     localDW->is_Rich_Mixture = Step_07_logic_IN_Single_Failure;
124     break;
125
=> 126     default:
127         localDW->is_Running = Step_07_logic_IN_NO_ACTIVE_CHILD;
128         break;
129     }
130 }
131 }
132 break;
133

```

OK Help

# Code Equivalence Check Results: Model vs Code

Test Manager interface showing test results. The 'RESULTS' tab is active, displaying a table of test results. A red box highlights the entry '02\_Functional\_Baseline\_Excel\_FullCoverage' with a status of '19' and a green checkmark.

NAME	STATUS
Results : 2015-Jun-04 22:45:56	19 ✓
02_Functional_Baseline_Excel_FullCoverage	19 ✓

Stateflow (chart) CruiseControl/Compute target speed - Simulink. The chart displays the logic for computing target speed, including states like OFF, CRUISE, and ON. A coverage window is overlaid on the chart, indicating full coverage for the 'Compute target speed' block.

Coverage: CruiseControl\_Harness\_Normal

SubSystem block "Compute target speed"

Full Coverage

Ready 81% FixedStepDiscrete

# Code Equivalence Check Results: Model vs Code

Test Manager

TESTS VISUALIZE FORMAT

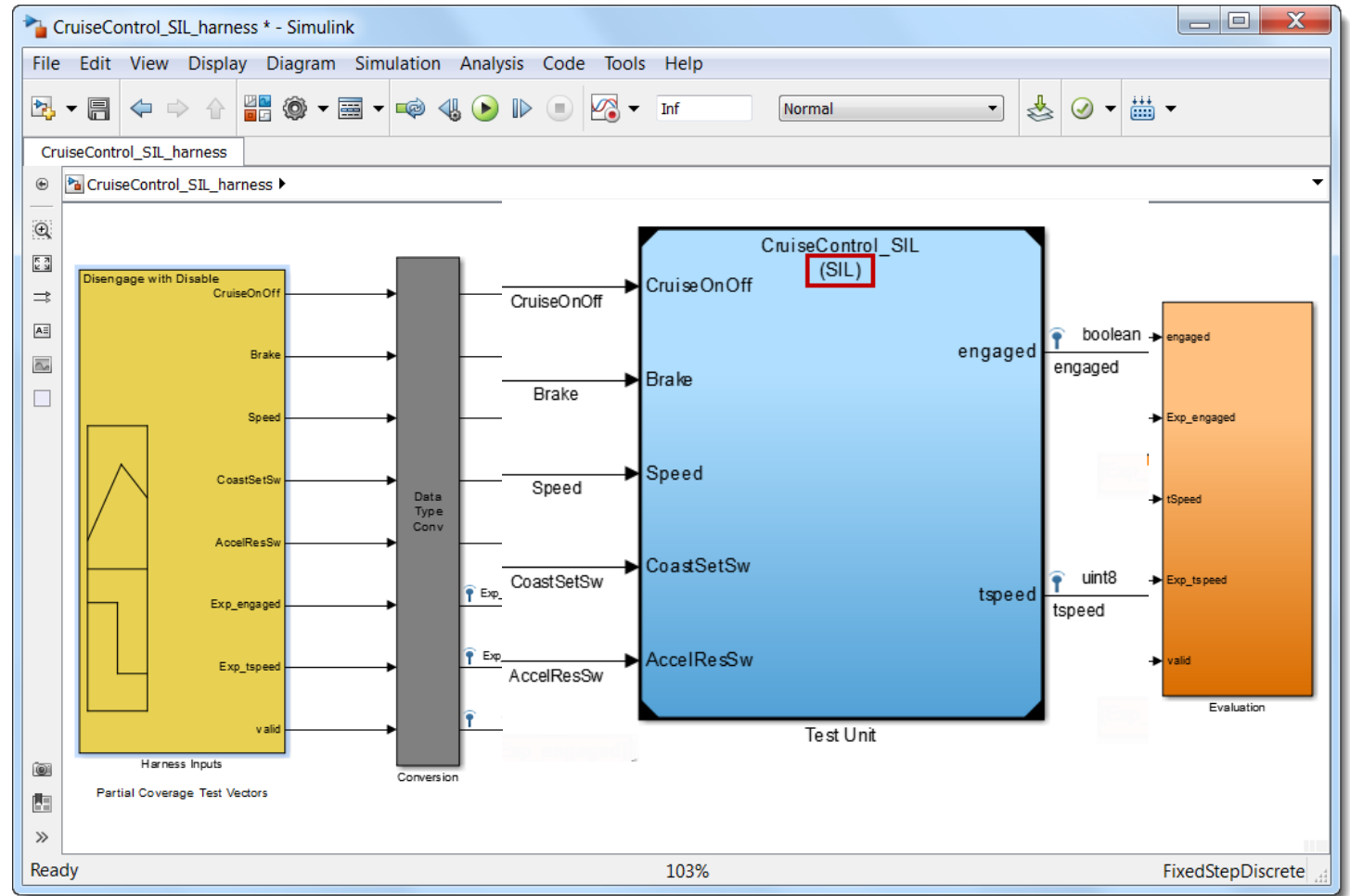
Clear Plot Data Cursors Highlight in Model Send to Figure

EDIT ZOOM & PAN MEASURE & TRACE SHARE

Test Browser Results and Artifacts

Filter Results

NAME	STATUS
Results : 2015-Jun-04 22:45:56	19 ✓
02_Functional_Baseline_Excel_FullCoverage	19 ✓
Results : 2015-Jun-04 23:01:48	19 ✓
03_Equivalence_Normal_SIL_Excel	19 ✓
Init_Not_Enable_or_Engaged	✓
Equivalence Criteria Result	✓
engaged	✓
tspeed	✓
AccelResSw	✓
Brake	✓
CoastSetSw	✓
CruiseOnOff	✓
Speed	✓
Sim Output 1(CruiseControl : normal)	
Sim Output 2(CruiseControl : normal)	
Not_Engaged_with_Enable	✓
Disengage_with_Disable	✓
Engage_1st_with_CoastSetSW	✓
No_Engage_1st_with_AccelResSW	✓
Disengage_with_Brake	✓
Re_Eng with CoastSetSw	✓



# Code Equivalence Check Results: Model vs Code

The screenshot shows the Test Manager interface with the 'Results and Artifacts' tab selected. A search filter is applied to 'Filter Results'. The test results table shows a successful run for '03\_Equivalence\_Normal\_SIL\_Excel' with 19 test cases passing. The test cases listed include 'Init\_Not\_Enable\_or\_Engaged', 'Equivalence Criteria Result', and various simulation outputs and conditions.

NAME	STATUS
Results : 2015-Jun-04 22:45:56	19 ✓
02_Functional_Baseline_Excel_FullCoverage	19 ✓
Results : 2015-Jun-04 23:01:48	19 ✓
03_Equivalence_Normal_SIL_Excel	19 ✓
Init_Not_Enable_or_Engaged	✓
Equivalence Criteria Result	✓
engaged	✓
tspeed	✓
AccelResSw	✓
Brake	✓
CoastSetSw	✓
CruiseOnOff	✓
Speed	✓
Sim Output 1(CruiseControl : normal)	
Sim Output 2(CruiseControl : normal)	
Not_Engaged_with_Enable	✓
Disengage_with_Disable	✓
Engage_1st_with_CoastSetSW	✓
No_Engage_1st_with_AccelResSW	✓
Disengage_with_Brake	✓
Re_Eng with CoastSetSw	✓

## Code Coverage

File Contents/Complexity	Cumulative			
	D1	C1	MCDC	Stmt
1. <a href="#">CruiseControl.c</a>	22 97%	98%	96%	100%
2. ... <a href="#">CruiseControl_Init</a>	1 --	--	--	100%
3. ... <a href="#">CruiseControl</a>	20 97%	98%	96%	100%
4. ... <a href="#">CruiseControl_initialize</a>	1 --	--	--	100%

```

47 /* Output and update for referenced model: 'CruiseControl' */
48 void CruiseControl(const boolean_T *rtu_CruiseOnOff, const boolean_T *rtu_Brake,
49                   const uint8_T *rtu_Speed, const boolean_T *rtu_CoastSetSw,
50                   const boolean_T *rtu_AccelResSw, boolean_T *rty_engaged,
51                   uint8_T *rty_tspeed)
52 {
53   /* Chart: '<Root>/Compute target speed' */
54   /* Gateway: Compute target speed */
55   if (CruiseControl_DW.temporalCounter_i1 < MAX_uint32_T) {

```

**Decisions analyzed:**

Decision	#1	#2	Total
CruiseControl_DW.temporalCounter_i1 < MAX_uint32_T	50%	0%	50%
false	0/399	--	0/420
true	399/399	--	420/420

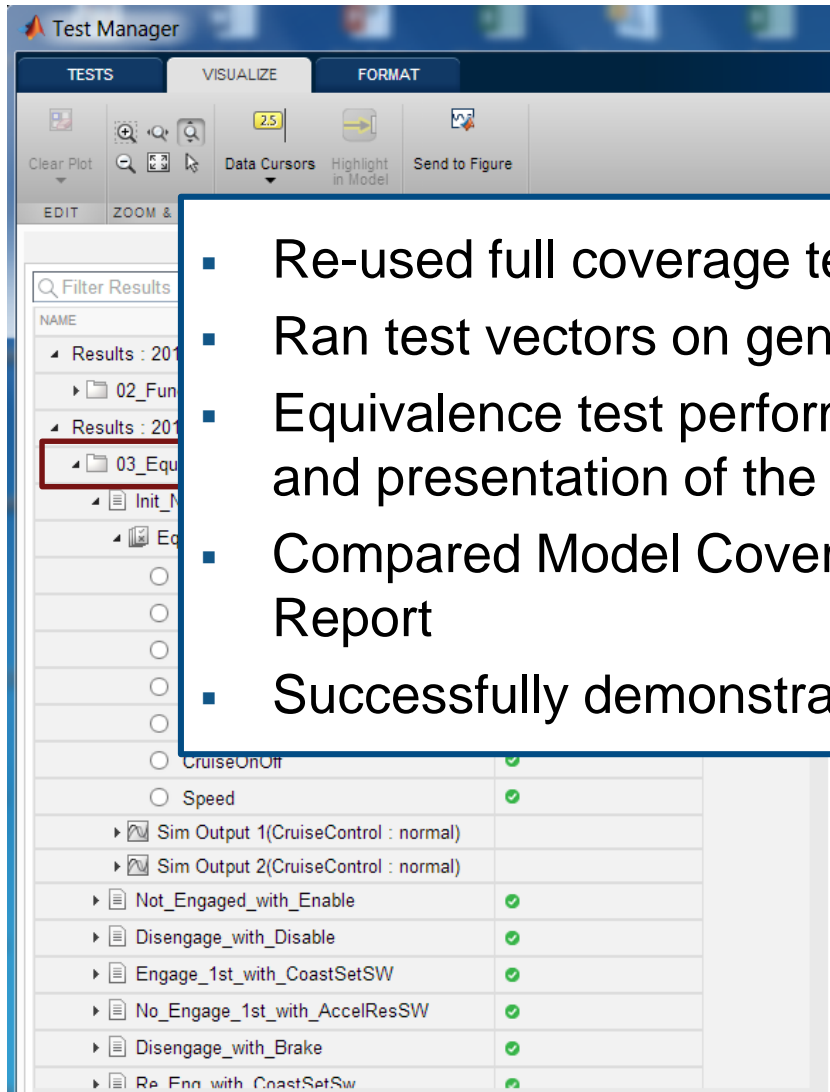
```

66   CruiseControl_DW.AccelResSw_prev = *rtu_AccelResSw;
67   CruiseControl_DW.CoastSetSw_prev = *rtu_CoastSetSw;
68

```

# Code Equivalence Check Results: Model vs Code

## Code Coverage



File Contents/Complexity	Cumulative			
	DI	CI	MCDC	Stmt
1. <a href="#">CruiseControl.c</a>	22 97%	98%	96%	100%
2. <a href="#">CruiseControl_Init</a>	1 --	--	--	100%
3. <a href="#">CruiseControl</a>	20 97%	98%	96%	100%

- Re-used full coverage test vectors and harnesses from Model Verification testing
- Ran test vectors on generated code using Model Reference SIL mode
- Equivalence test performed in Simulink Test, including test execution, evaluation and presentation of the results
- Compared Model Coverage to Code Coverage using the SIL Code Coverage Report
- Successfully demonstrated code behavior matches model behavior!

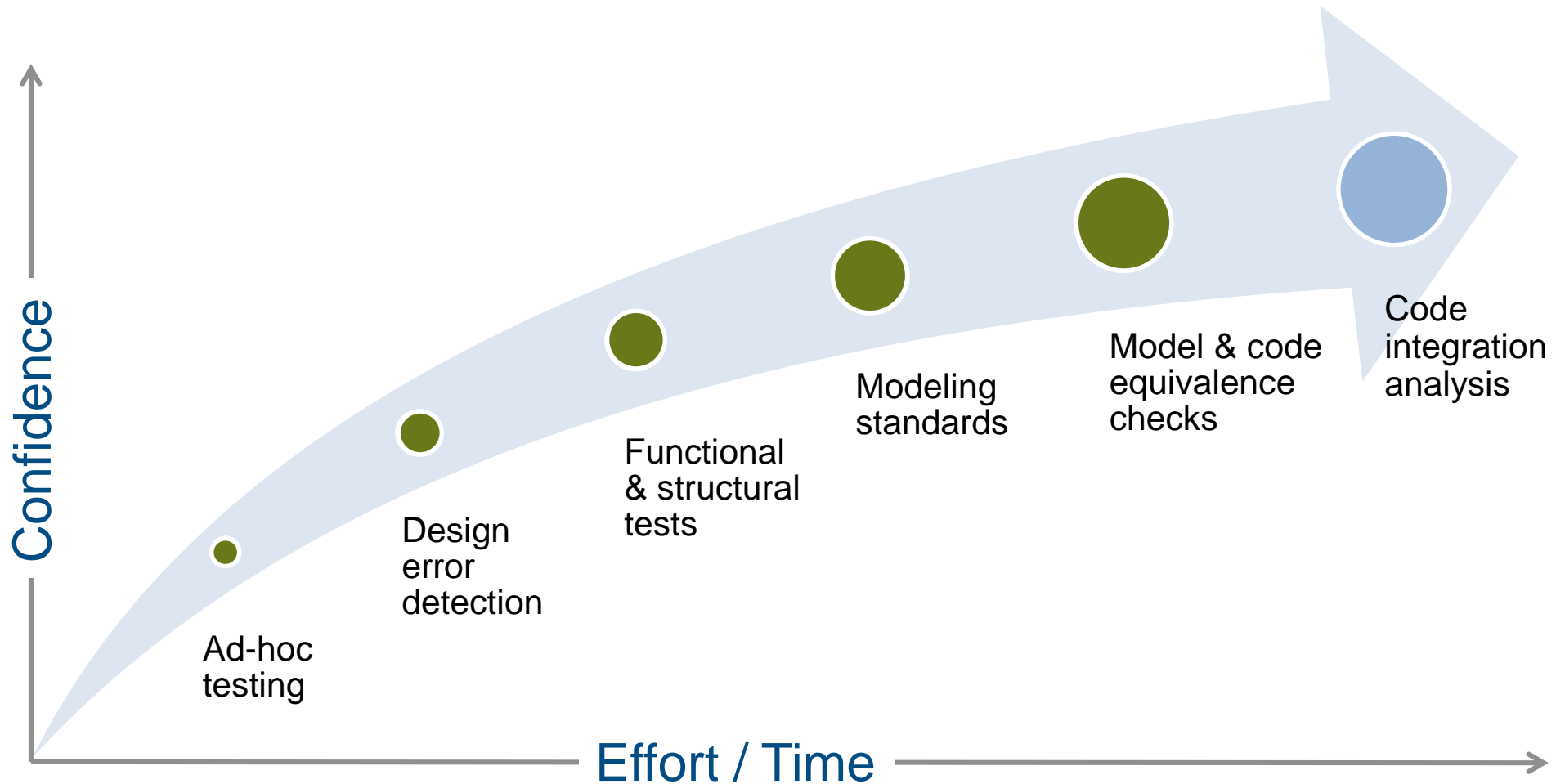
	#1	#2	Total
CruiseControl_DW.temporalCounter_i1 < MAX_uint32_T	50%	0%	50%
false	0/399	--	0/420
true	399/399	--	420/420

```

66 CruiseControl_DW.AccelResSw_prev = *rtu_AccelResSw;
67 CruiseControl_DW.CoastSetSw_prev = *rtu_CoastSetSw;
68

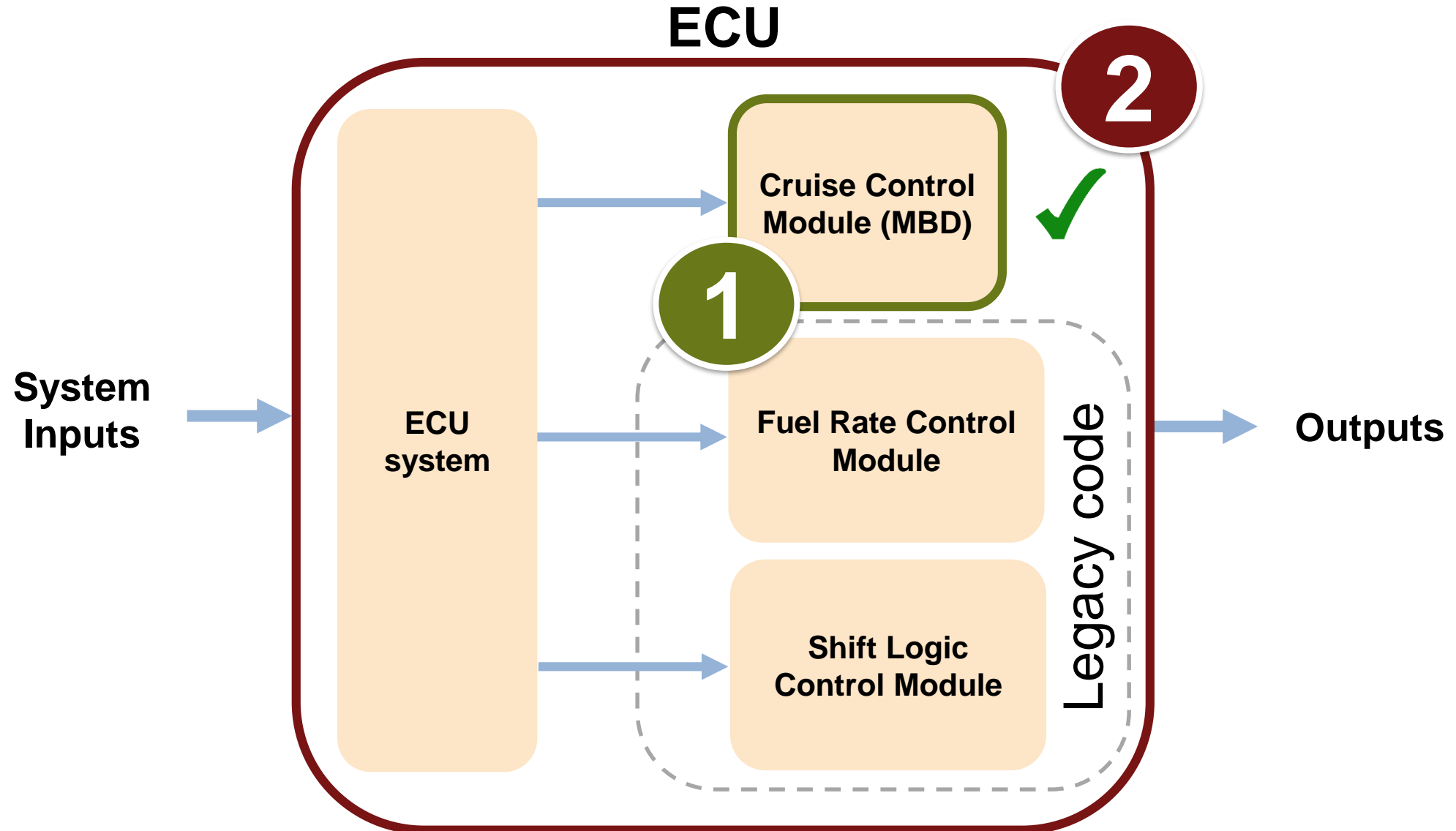
```

# Gaining Confidence in our Design

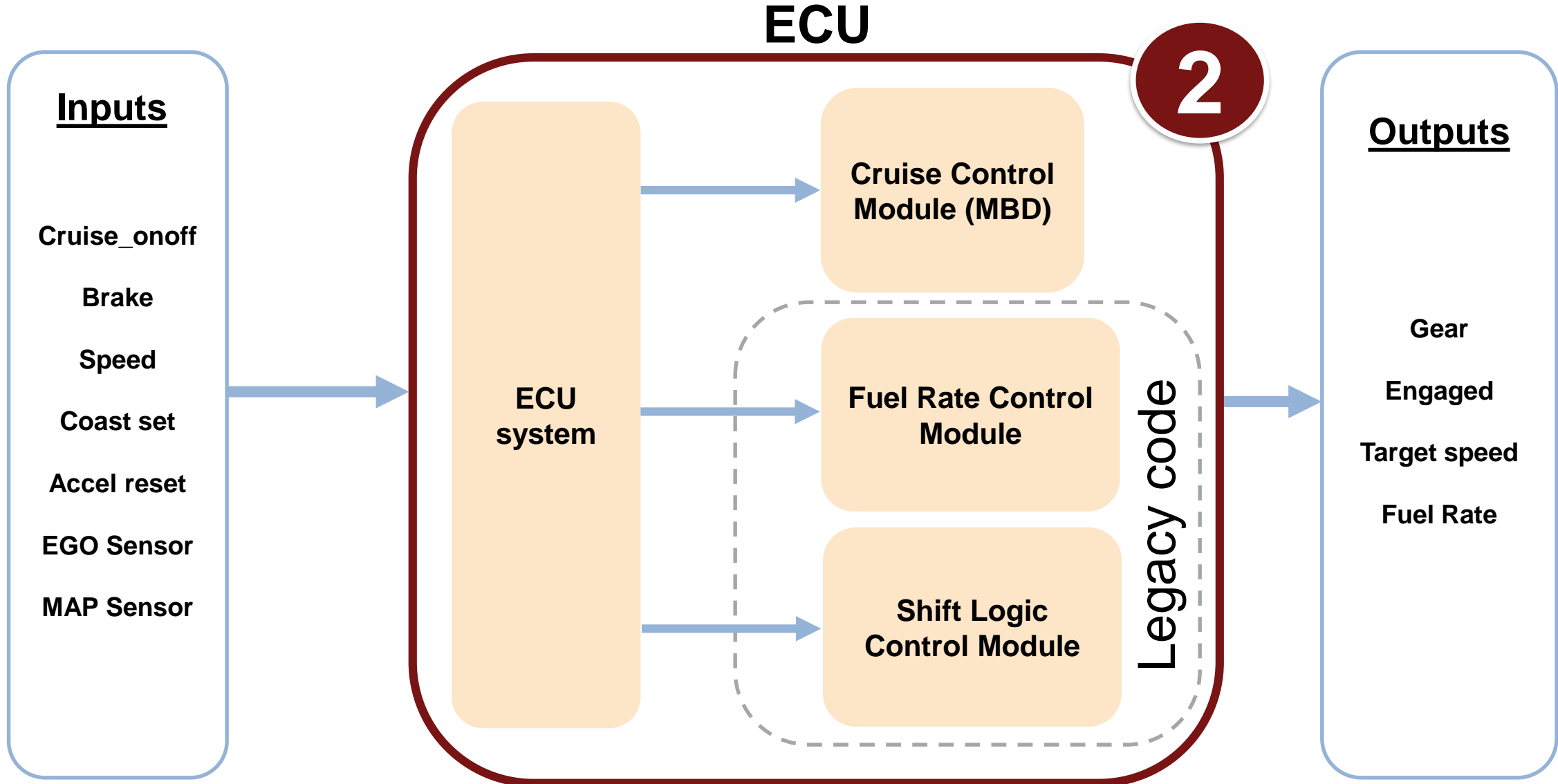




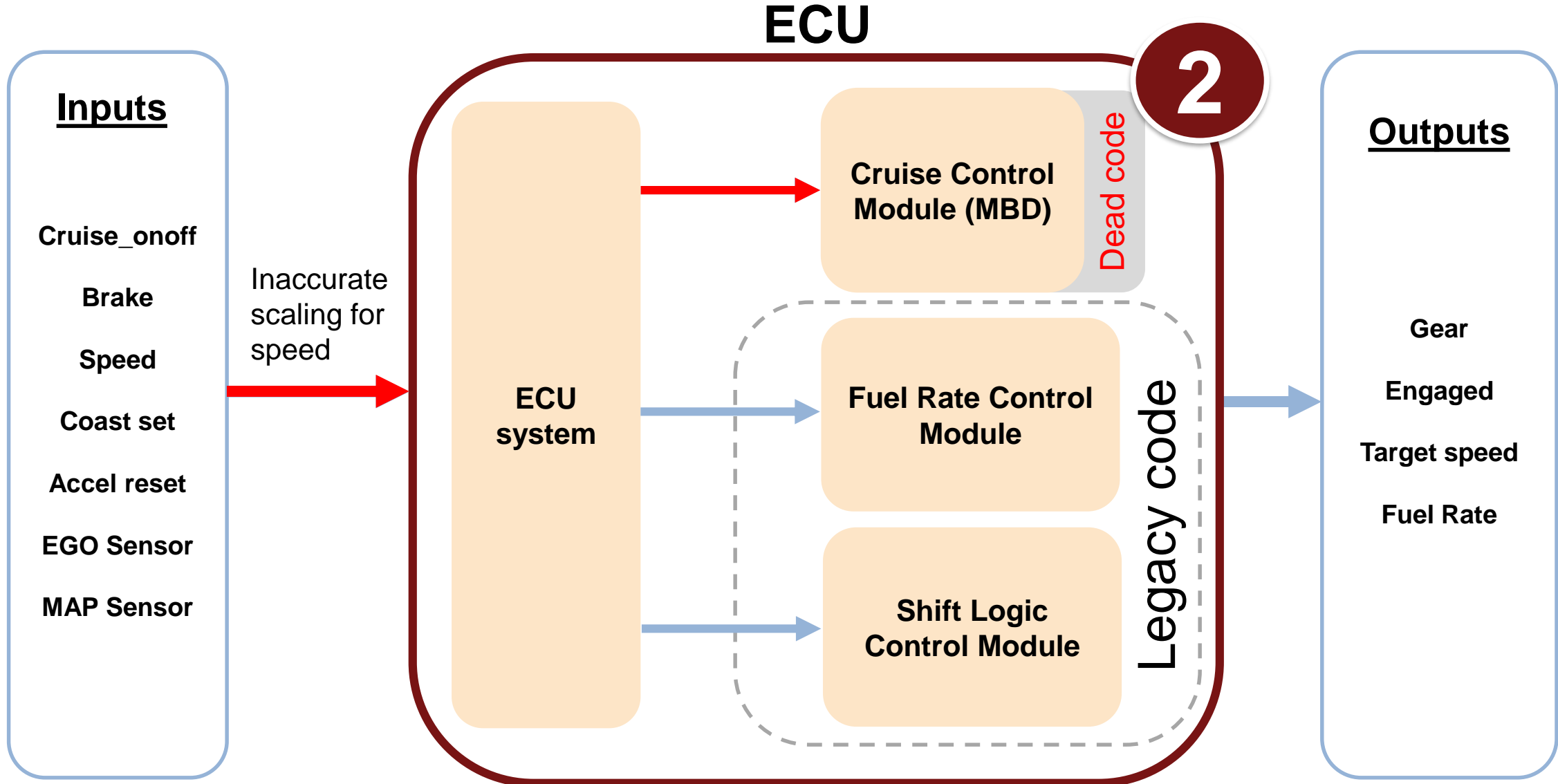
# Code Integration Analysis



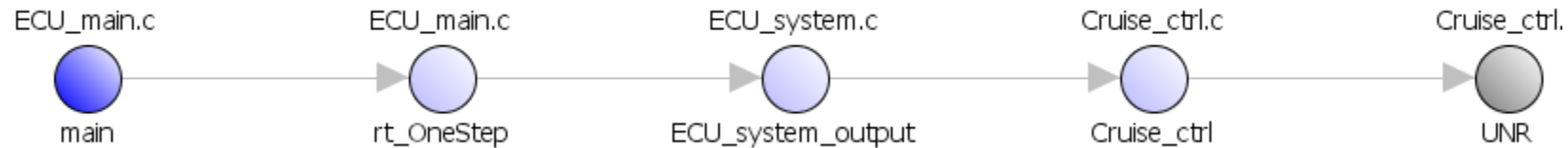
# Code Integration Analysis



# Finding Dead Code During Integration



# Finding Dead Code with Polyspace



Target speed parameter  
propagated to “Cruise\_ctrl.c”  
[0 ... 40]

Maximum target speed = 90

```

/* Entry 'STANDBY': '<S5>:52' */
*rtv Engaged = false;
} else if (rtu Speed > maxtspeed) {
/* Transition: '<S5>:55' */
/* Exit Internal 'ON': '<S5>:54' */
localDW->is ON = IN_NO_ACTIVE_CHILD;
localDW->is CRUISE = IN_STANDBY;

/* Entry 'STANDBY': '<S5>:52' */
*rtv Engaged = false;
} else if (rtu Speed < mintspeed) {
/* Transition: '<S5>:113' */

```

Dead code

# Root Cause for Dead Code: Speed Sensor Input Hand Code

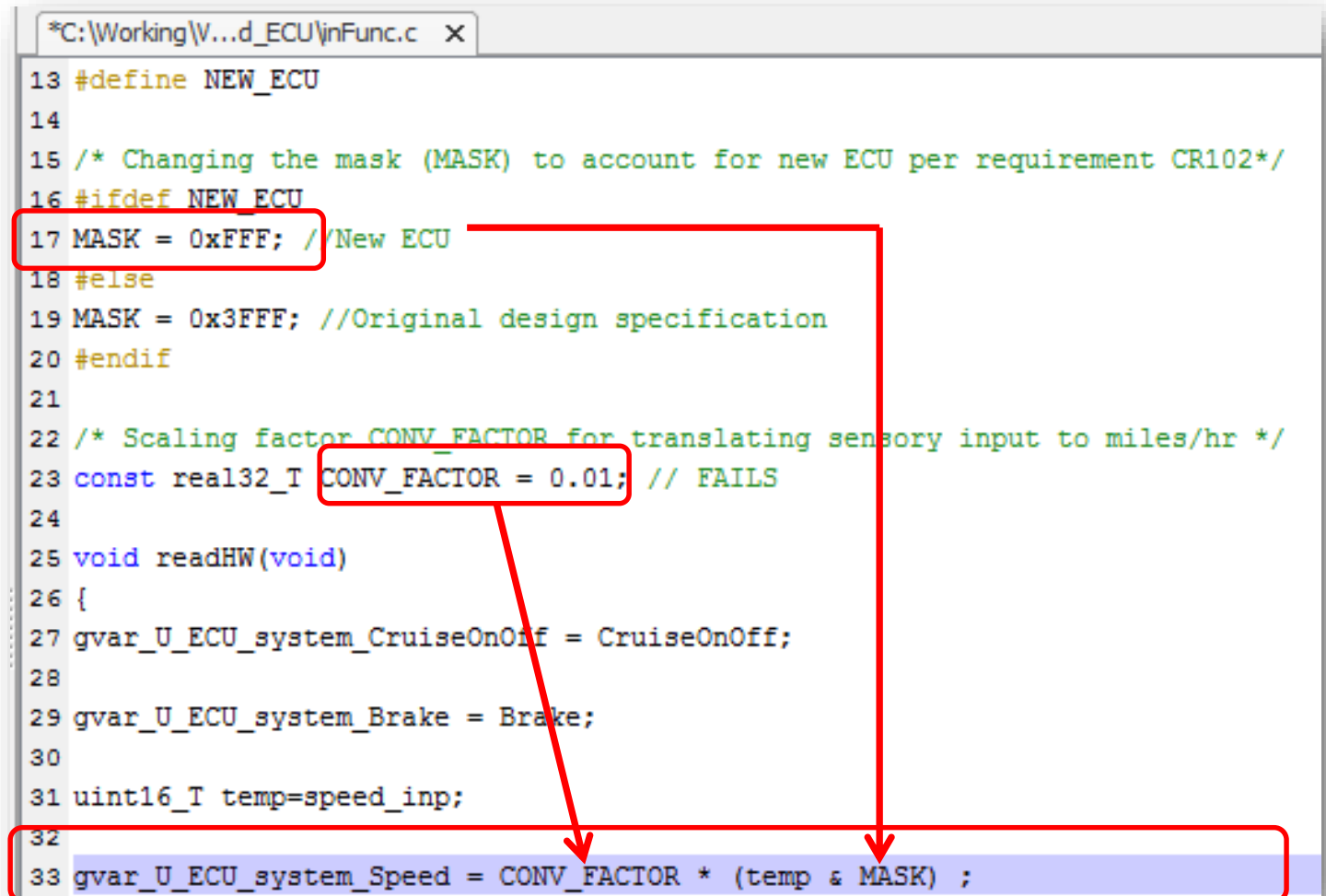
Changing analog-to-digital converter from 14 to 12-bit results in dead code

MASK – accounts for scaling down for new ADC from 14-bit to 12-bit

CONV\_FACTOR – accounts for translating sensor input counts to mph

Overlooked changing CONV\_FACTOR for new ADC

```
*C:\Working\V...d_ECU\inFunc.c x
13 #define NEW_ECU
14
15 /* Changing the mask (MASK) to account for new ECU per requirement CR102*/
16 #ifdef NEW_ECU
17 MASK = 0xFFF; //New ECU
18 #else
19 MASK = 0x3FFF; //Original design specification
20 #endif
21
22 /* Scaling factor CONV_FACTOR for translating sensory input to miles/hr */
23 const real32_T CONV_FACTOR = 0.01; // FAILS
24
25 void readHW(void)
26 {
27 gvar_U_ECU_system_CruiseOnOff = CruiseOnOff;
28
29 gvar_U_ECU_system_Brake = Brake;
30
31 uint16_T temp=speed_inp;
32
33 gvar_U_ECU_system_Speed = CONV_FACTOR * (temp & MASK) ;
```



# Polyspace Code Analysis

Start with C/C++ source code

```
static void pointer_arithmetic (void) {
    int array[100];
    int *p = array;
    int i;

    for (i = 0; i < 100; i++) {
        *p = 0;
        p++;
    }

    if (get_bus_status() > 0) {
        if (get_oil_pressure() > 0) {
            *p = 5;
        } else {
            i++;
        }
    }
}

i = get_bus_status();

if (i >= 0) {
    *(p - i) = 10;
}
}
```

# Polyspace Code Analysis

Source code painted in green, red, gray, orange

**Green: reliable**  
safe pointer access

**Red: faulty**  
out of bounds error

**Gray: dead**  
unreachable code

**Orange: unproven**  
may be unsafe for some conditions

**Purple: violation**  
MISRA-C/C++ or JSF++  
code rules

**Range data**  
tool tip

```

static void pointer_arithmetic (void) {
    int array[100];
    int *p = array;
    int i;

    for (i = 0; i < 100; i++) {
        *p = 0;
        p++;
    }

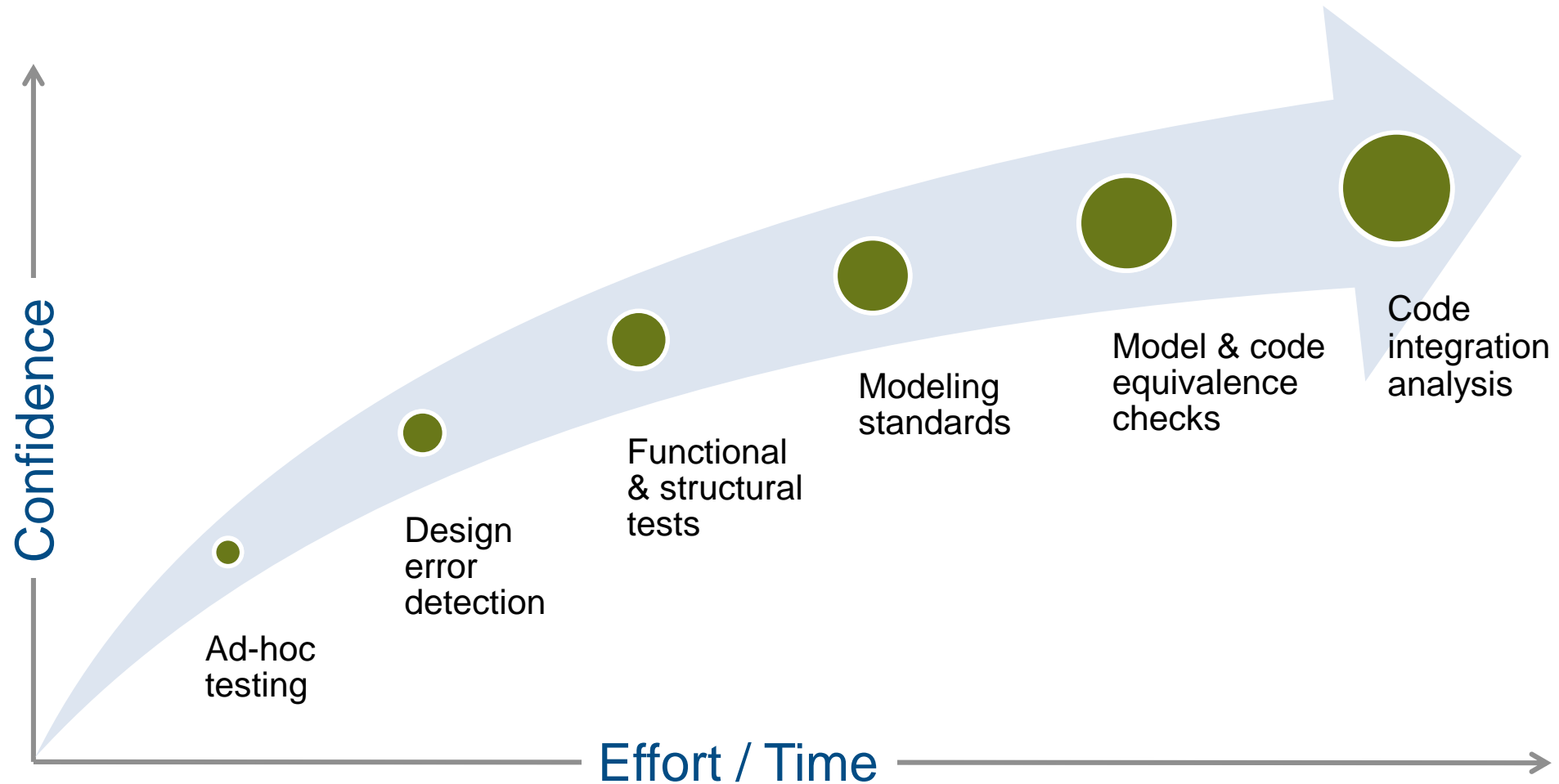
    if (get_bus_status() > 0) {
        if (get_oil_pressure() > 0) {
            *p = 5;
        } else {
            i++;
        }
    }

    i = get_bus_status();

    if (i >= 0) {
        *(p - i) = 10;
    }
}
    
```

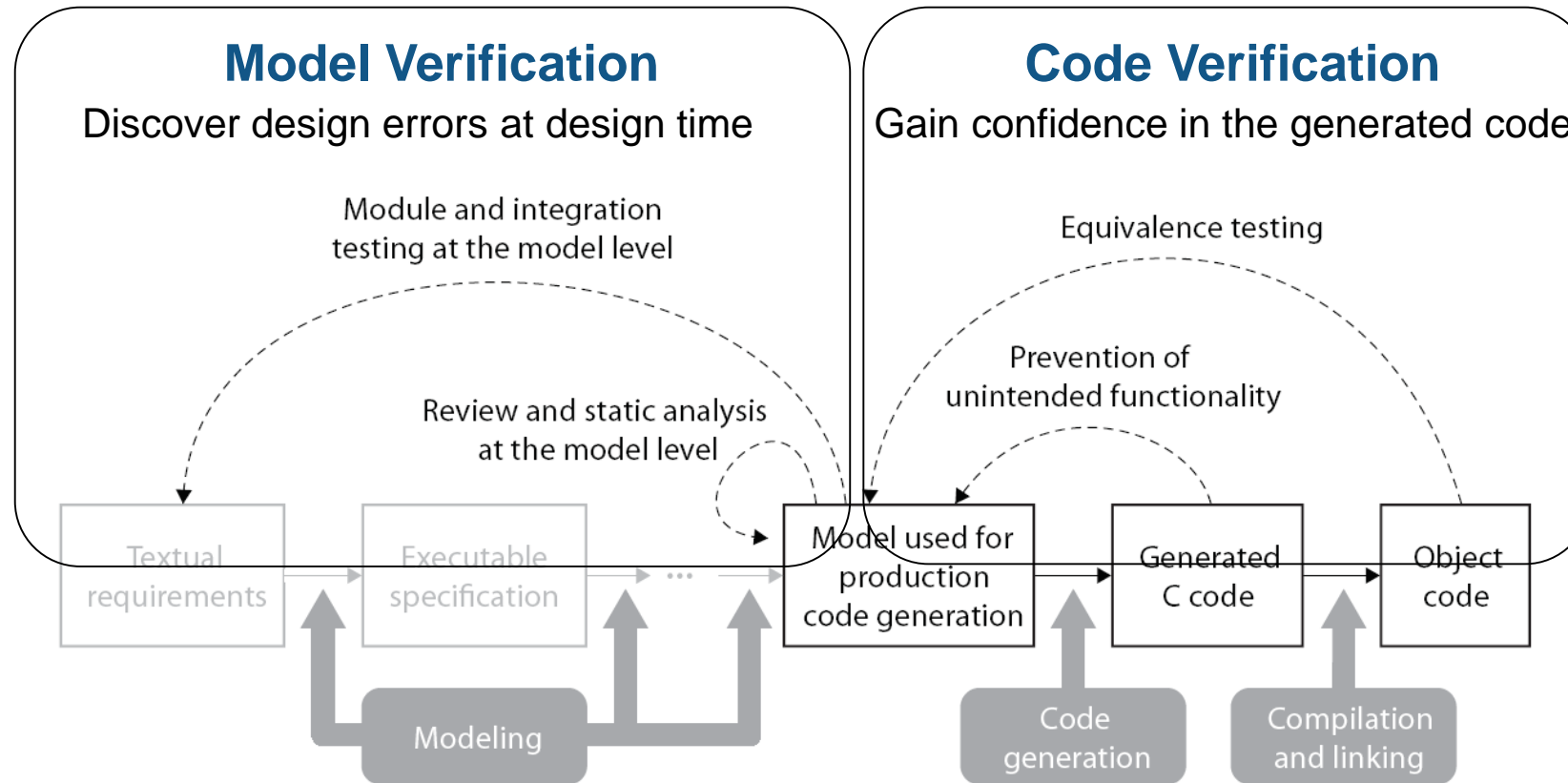
variable 'i' (int32): [0 .. 99]  
assignment of 'i' (int32): [1 .. 100]

# Gaining Confidence in our Design





# Conclusion: Model-Based Design Verification Workflow



*Workflow approved by TÜV SÜD for development of safety-critical software in accordance with ISO 26262 (automotive), IEC 61508 (industrial), EN 50128 (railway), IEC 62304 (medical devices)*

# Conclusion

*It is easier and less expensive to fix design errors early in the process when they happen.*

## **Model-Based Design enables:**

1. *Early testing to increase confidence in your design*
2. *Delivery of higher quality software throughout the workflow*



Change the world by

# Accelerating the pace

of discovery, innovation, development, and learning

# in engineering and science