Utilization of Simulink Verification and Validation (V&V) and Simulink Design Verifier (SDV) for HVAC Controls Software

Arun Chakrapani Rao, Rupesh Kakade, Mohan Murugesan
HVAC Controls Algo And Readiness, Electronic Controls and Software Engineering,
General Motors Technical Centre India Pvt. Ltd., Bengaluru, India

Outline

- Readiness Testing and Core Algorithm work overview
- HVAC Production-oriented testing (ECU, model)
- What is structural coverage? Why use it?
- What are model coverage metrics?
- Overview of work done and results
- Recommendations for incremental improvements
- Potential for Automatic Test case Generation
- Potential for Property Proving
- Current challenges and some proposed workflows

HVAC Control Software

Regulates the ***air temperature, flow rate and moisture***

throughout the vehicle interior (by considering the effects of

ambient temperature, sun load, and heat transfer mechanisms)

in **real-time**

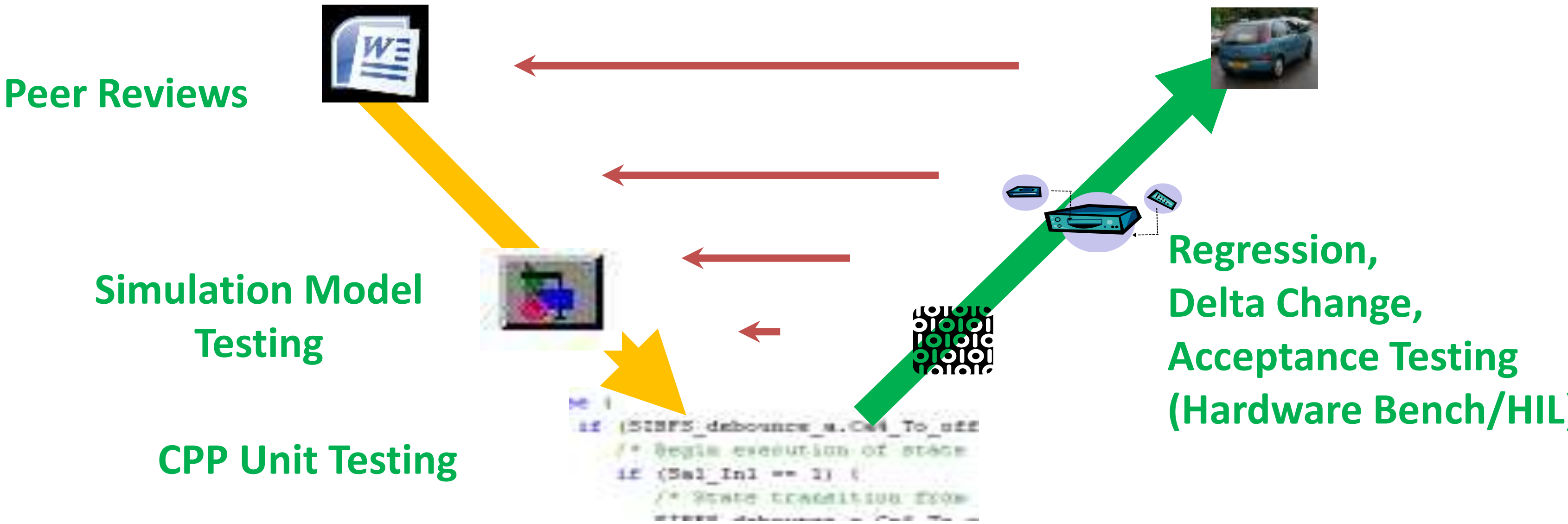
Challenges overcome using Model-Based Designs in Development

- Unit level and integrated software verified early
- Same software deployed to many different vehicles by simply calibrating parameters such as vehicle dimensions
- Same s/w also deployed to multiple controllers with varying hardware and software architecture (Non-standard or standard ones like AUTOSAR)
- Integration of legacy software and the model-based software possible for vehicles nearing production
- Parallel development of several components possible
- Production code auto-generated, compiled and targeted efficiently and accurately

HVAC Control Software – Example Components

- **Aero Shutter Control**
  - Combinational logic for on/off control of magnetically driven set of flaps which close front end airflow paths to enhance vehicle aerodynamics
- **Cabin Air Recirculation Control**
  - Physics-based design to ensure minimal compressor work while maintaining thermal comfort of the occupants
  - Repeated calculations (physical properties) implemented by creating and using our own library blocks
  - Functional verification using approximate plant model for closed-loop simulation
  - Standard test inputs derived from requirements and vehicle like scenarios (vehicle test data)

Current Testing in Production



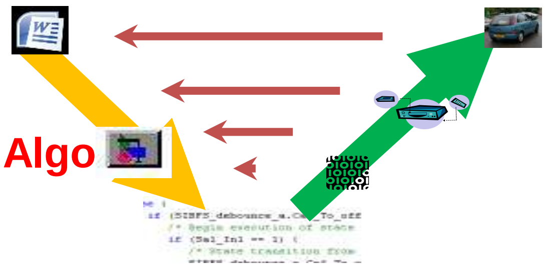
Test cases mainly guided by **Requirements**

Both **Manual** and **Automated Testing**

Core Algorithm Modeling Group

Development and testing of various HVAC component models

Core Algo



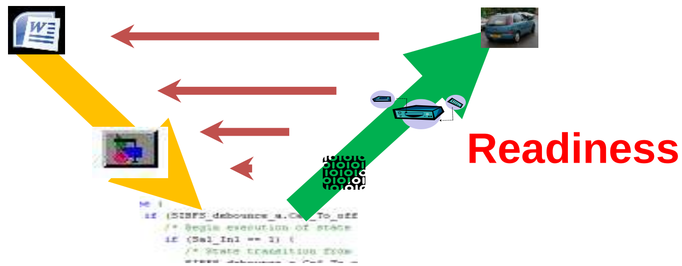
- **Simulation Model Testing**
  - Performed at the unit level
  - Closed-loop simulation of the control system with approximate plant model
  - Detailed functional verification based on requirements, internal standards and over several vehicle like scenarios
  - Performed using standard test inputs developed once

- **CPP Unit Testing**
  - Simulation model I/Os are automatically translated using a MATLAB M-script
  - Verifies interface between the automatically generated code from the model and the wrapper interface code and the buried conversion mathematics
  - Performs acceptance check for example, requirements, rounding errors etc. with the use of CPP asserts

Plant models for closed-loop simulation
Simulation and early verification possible

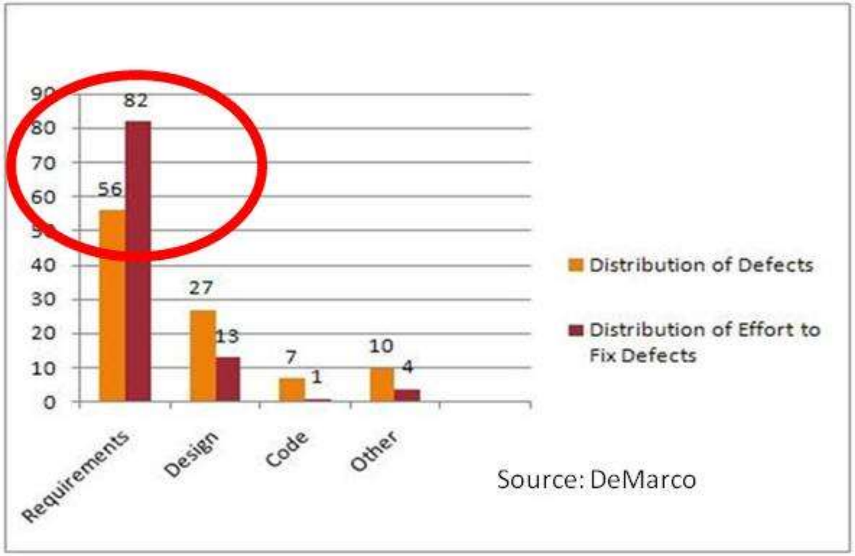
Readiness Group

Testing of
HVAC components
at the
integrated ECU



- **Regression Test**
  - Detailed Component level verification
  - Performed once on a Model Year Software
  - Performed using automated test scripts on dSPACE HIL
- **Delta Change Verification**
  - Verifies the specific delta change on every release
  - Manual / automated test scripts
- **Acceptance Test**
  - Verifies the system level functionalities on every release
  - Performed using automated test scripts on dSPACE HIL

Shift towards early model-based V&V



Phase in which defect gets fixed	Relative cost
Requirements	1
Design	3 – 6
Coding	10
Development Testing	15 – 40
Acceptance Testing	30 – 70
Operations	40 – 1000

Structural Coverage

The output shall be set to 100 times the sensor input.

$-5 \leq I_1 \leq 40$

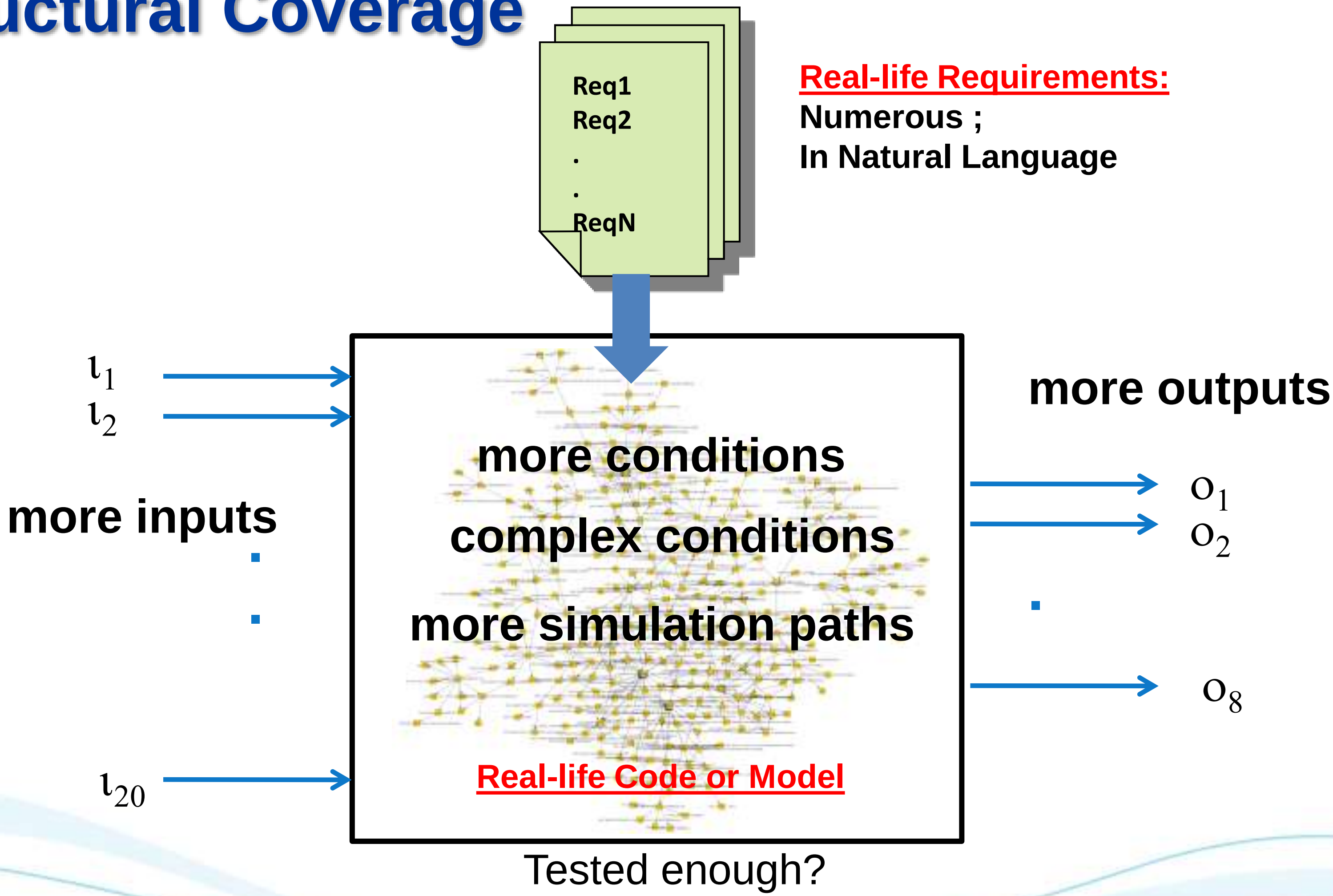
Sample inputs: -15, 10, 45
Boundary values: -5, -4.9, 40, 40.1

If sensor input is valid, the output shall be 100 times, else a fail safe value of 180 should be output.

Valid and invalid cases also

Choices of input values affect the calculations done downstream
Overall coverage gets influenced by such choices!

Structural Coverage



Irrespective of the test design techniques, in real-life scenario,

model coverage assessment becomes necessary and crucial!

Why Structural Coverage?

- Find out gaps in requirements-based test cases
- Identify gaps in requirements
- Identify unreachable parts of the model (or code)
- Identify unintended functionality

ISO/FDIS 26262-6:2010(E)

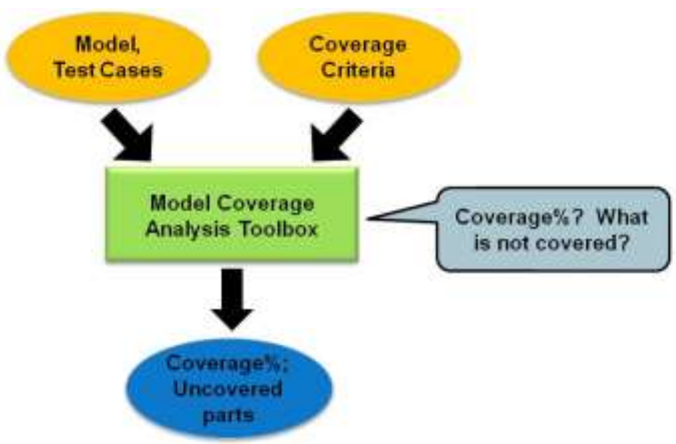
Table 12 — Structural coverage metrics at the software unit level

Methods		ASIL			
		A	B	C	D
1a	Statement coverage	++	++	+	+
1b	Branch coverage	+	++	++	++
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++

NOTE 2 In the case of model-based development, the analysis of structural coverage can be performed at the model level using analogous structural coverage metrics for models.

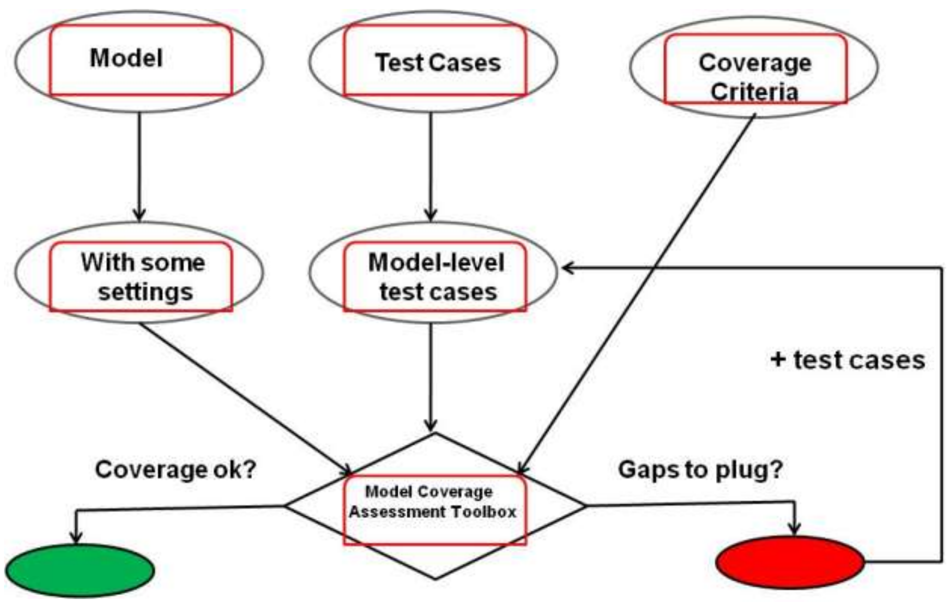
Structural Coverage Assessment

Principle



Ok with result – Document any justifications
Not Ok – Add more test cases to cover uncovered parts (manually/ATG)

Practice for Production

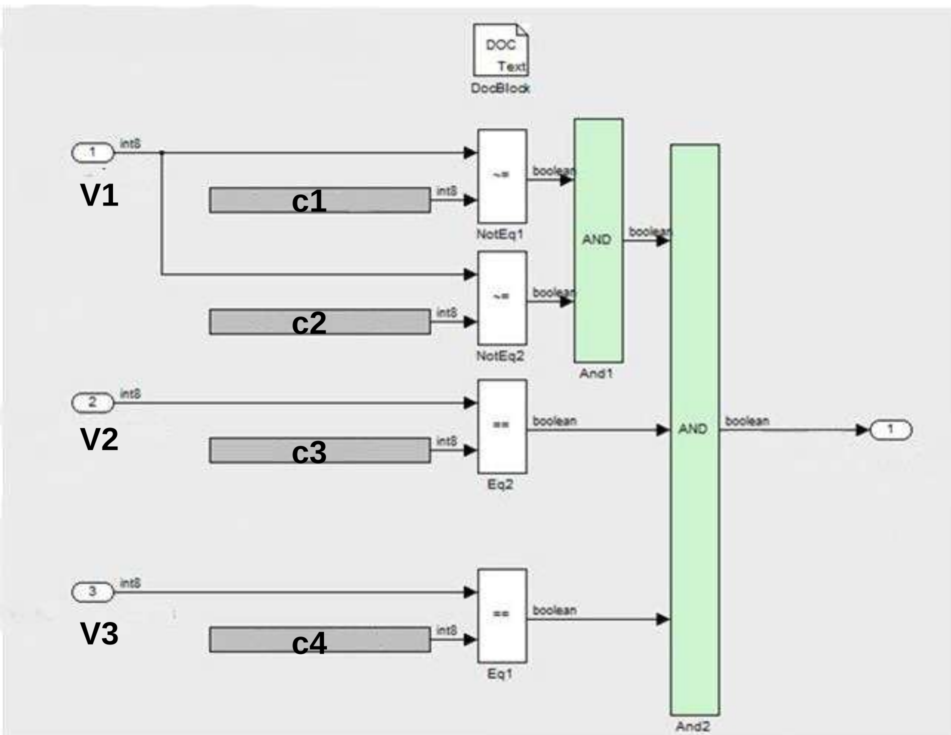


Relevant Mathworks toolbox:

Simulink Verification and Validation toolbox (V&V toolbox)

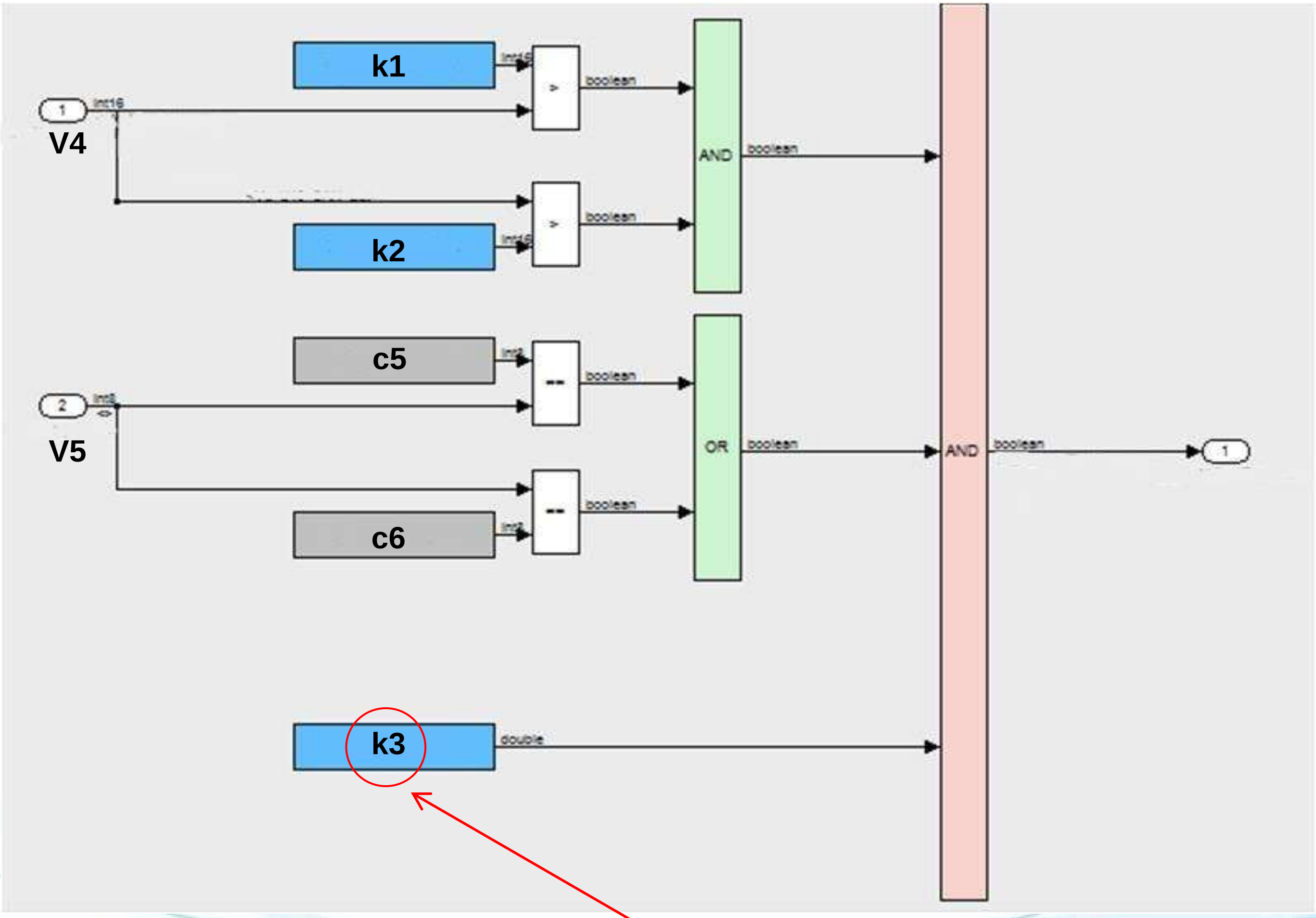
Model Coverage Metrics – Condition Coverage

- Condition Coverage
  - Analyzes blocks that output logical combinations of their inputs
  - Logical Operator blocks, Stateflow transitions



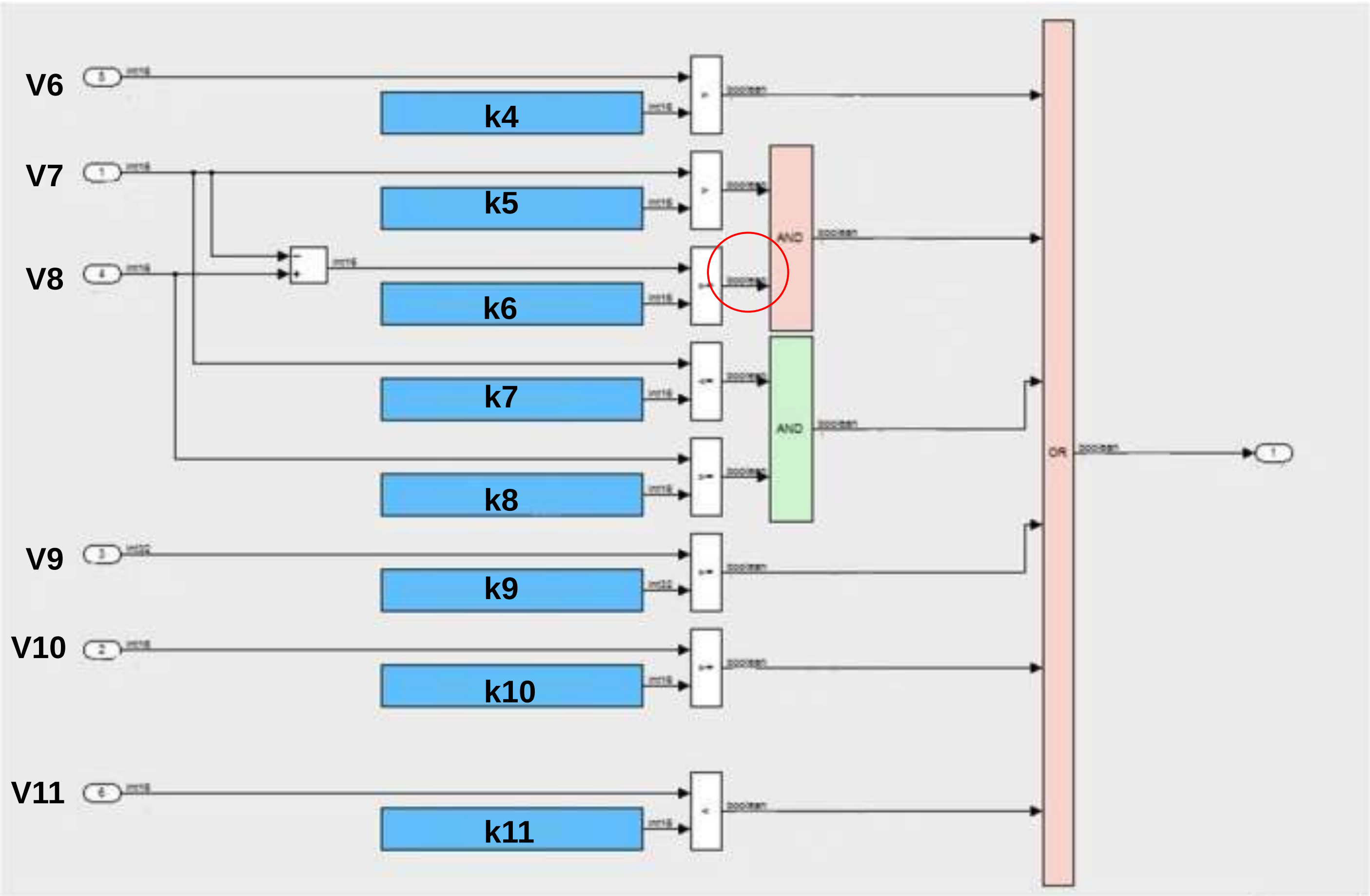
2 AND blocks;
2*2, 3*2

Model Coverage Metrics – Condition Coverage



Cal value was T in all test cases

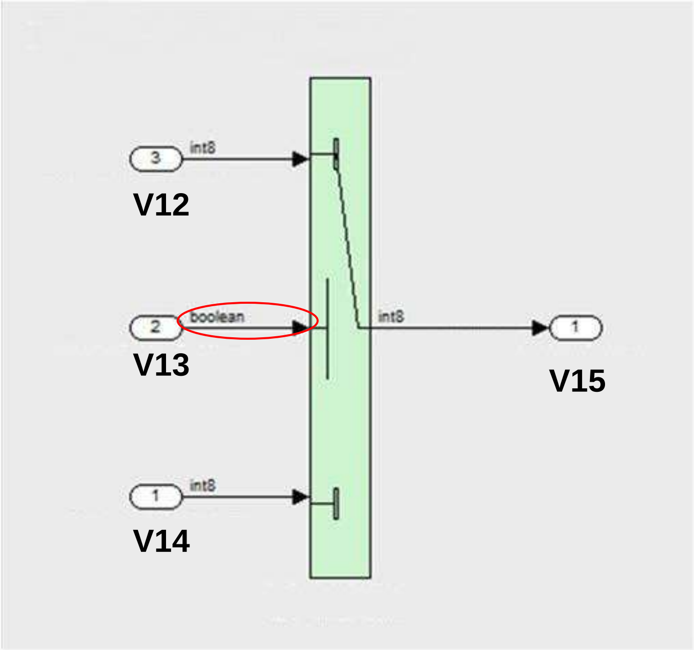
Model Coverage Metrics – Condition Coverage



No True for one of the AND conditions
=> making it T will cover 2 more conditions (for the AND, OR together)

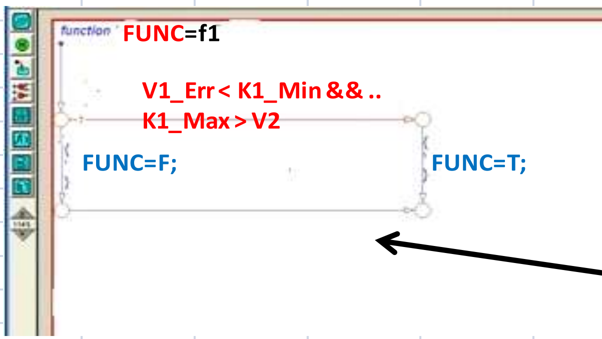
Model Coverage Metrics

- Decision Coverage
  - Analyzes model elements that represent decision points
  - Switch block, Stateflow states



Model Coverage Metrics

- MCDC
  - Independence of logical block inputs and transition conditions



INIT -> NORMAL

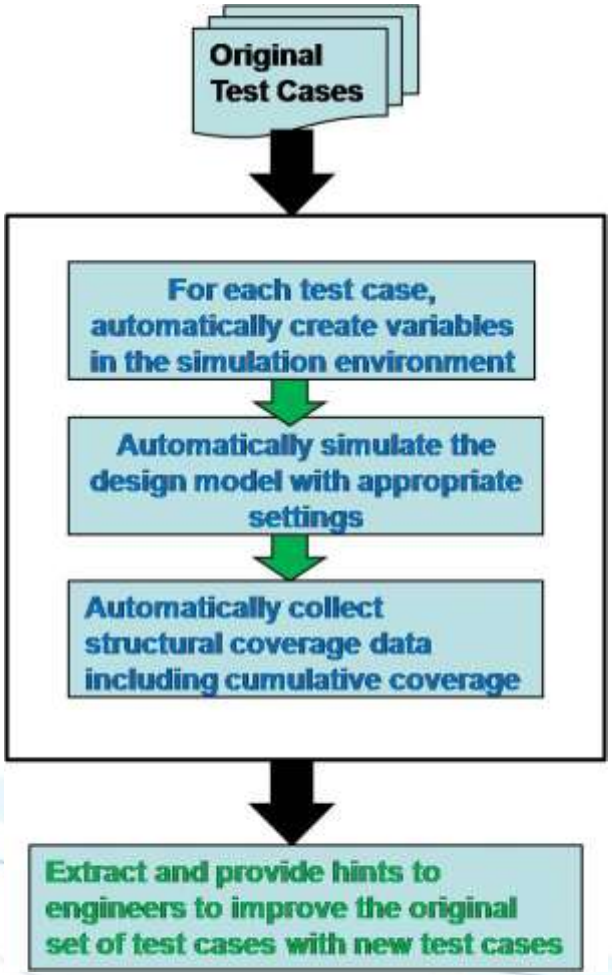
```
/* T1_2 */
[(Mode1 != C1) &&...
(Mode2 != C2) && ...
(Mode2 != C3) && ...
( (V1_Err > 0) || ...
f1() || ..
!f2() )]
```

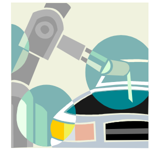
CC	**75% (3/4)**	
		T / F
C1	V1_Err < K1_Min	
C2	K1_Max > V2	
DC	**100% (2/2)**	
MCDC	**50% (1/2 conditions reveresed the outcome)**	
C1	V1_Err < K1_Min	
C2	K1_Max > V2	
Out	C1 && C2	

	T Out	F Out
C1	**TT**	**Fx**
C2	**TT**	**(TF)**

Stateflow Graphical Function with a condition of the form **C1 && C2**

Overview of automation done around V&V toolbox

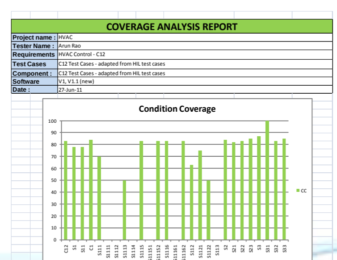


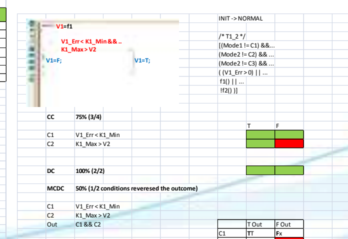


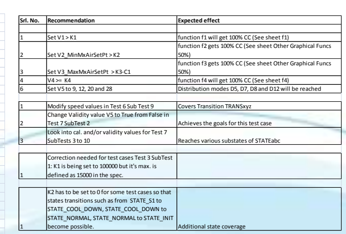
Internal tool for test automation

Excel sheet textual description of steps

MATLAB M Scripts for automation around the utilization of the **Simulink V&V toolbox** for structural coverage assessment



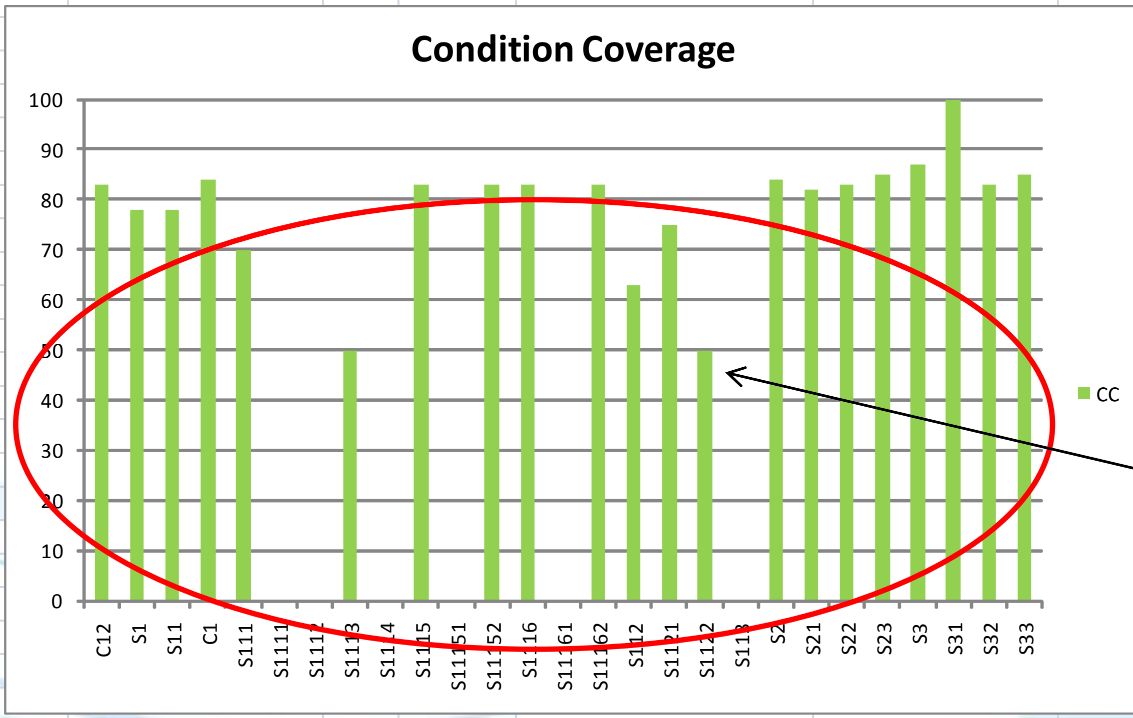




Recommendations to improve test cases

Report – Overview sheet

COVERAGE ANALYSIS REPORT	
Project name :	HVAC
Tester Name :	Arun Rao
Requirements	HVAC Control - C12
Test Cases	C12 Test Cases - adapted from HIL test cases
Component :	C12 Test Cases - adapted from HIL test cases
Software	V1, V1.1 (new)
Date :	27-Jun-11



Low coverage here!

Recommendations

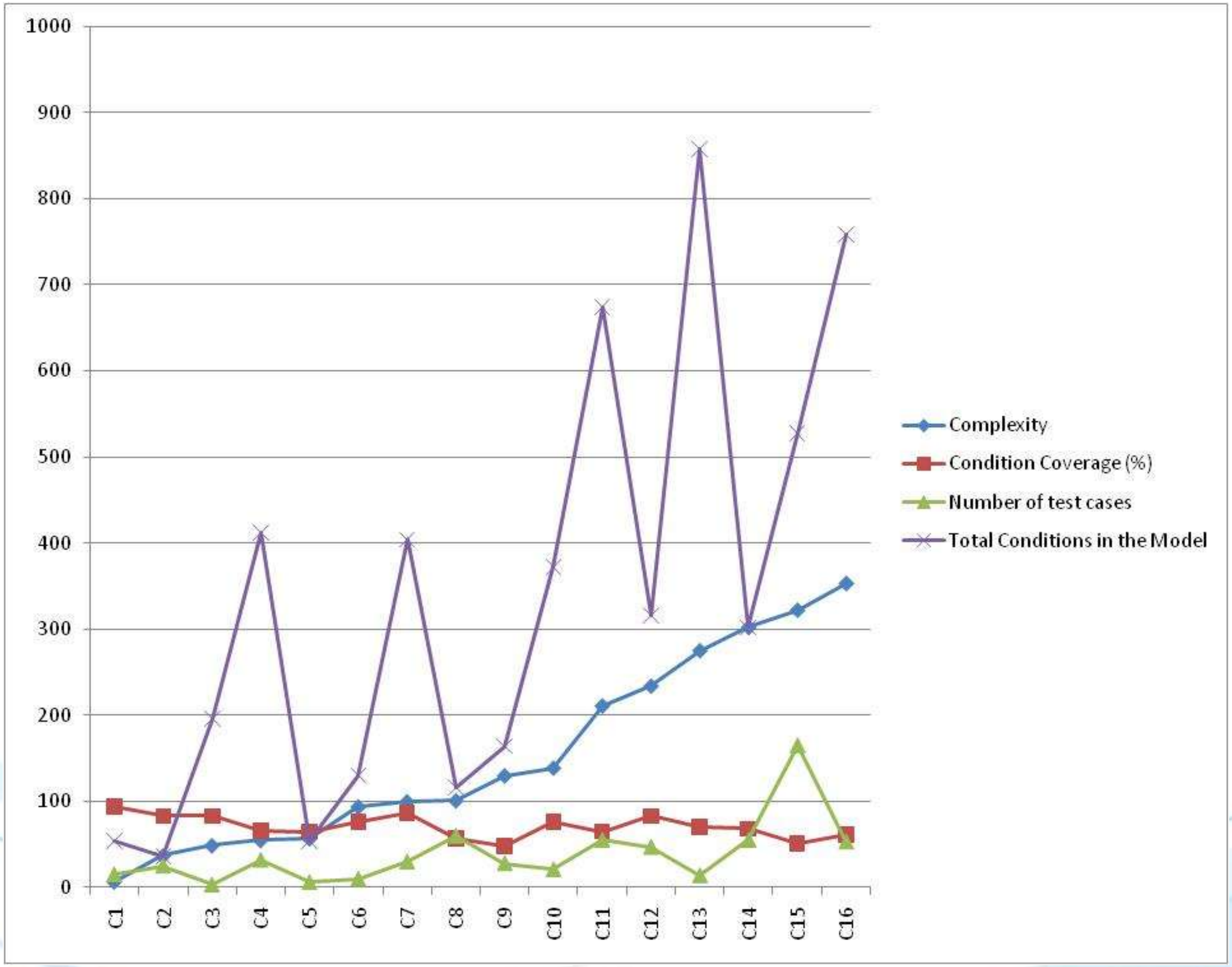
Sample recommendations for C12, C8, C2, C1

Srl. No.	Recommendation	Expected effect
1	Set V1 > K1	function f1 will get 100% CC (See sheet f1)
2	Set V2_MinMxAirSetPt > K2	function f2 gets 100% CC (See sheet Other Graphical Funcs 50%)
3	Set V3_MaxMxAirSetPt > K3-C1	function f3 gets 100% CC (See sheet Other Graphical Funcs 50%)
4	V4 >= K4	function f4 will get 100% CC (See sheet f4)
6	Set V5 to 9, 12, 20 and 28	Distribution modes D5, D7, D8 and D12 will be reached
1	Modify speed values in Test 6 Sub Test 9	Covers Transition TRANSxyz
2	Change Validity value V5 to True from False in Test 7 SubTest 2	Achieves the goals for this test case
3	Look into cal. and/or validity values for Test 7 SubTests 3 to 10	Reaches various substates of STATEabc
1	Correction needed for test cases Test 3 SubTest 1: K1 is being set to 100000 but it's max. is defined as 15000 in the spec.	
1	K2 has to be set to 0 for some test cases so that states transitions such as from STATE_S1 to STATE_COOL_DOWN, STATE_COOL_DOWN to STATE_NORMAL, STATE_NORMAL to STATE_INIT become possible.	Additional state coverage

Coverage for various components

Srl. No.	Component	Ver	Total test cases	CC	Cyclomatic Complexity	Total Conditions in the Model	Conditions Covered by Test Cases
1	C1	v2	15	94	7	54	51
2	C2	v1	25	83	37	36	30
3	C3	v2	3	83	49	196	162
4	C4	v2	32	66	55	412	270
5	C5	v2	6	65	57	54	35
6	C6	v2	10	76	94	130	99
7	C7	v2	30	86	100	404	346
8	C8	v1	60	57	101	116	66
9	C9	v2	28	48	130	164	78
10	C10	v1	21	76	139	372	283
11	C11	v2	55	65	211	674	437
12	C12	v1	47	83	234	316	262
13	C13	v2	14	70	275	858	604
14	C14	v2	55	68	302	302	204
15	C15	v2	165	51	322	528	268
16	C16	v1	53	61	353	758	460

Coverage for various components



Some learnings – Simulink V&V toolbox

- Original test cases created for the hardware bench/HIL
- Extra effort to recreate test cases; capture intention of the tester
- Solution for the future: Model-level test cases to be updated/created/maintained for Readiness testing
- Utilization of the results requires some extra effort and time from component owners
- Ideally suited for independent V&V activities to assist Production work and teams initially

Some key take-always

- Some components might have a very good coverage already
  - > 80% Condition Coverage
  - Small models/low complexity: C1, C2, C3
  - Test cases have evolved well over time: C7, C12
- Some components have lower coverage
  - Only around (50%-60%)
  - Larger models/higher complexity
  - Much large number of test cases also haven't helped; so, gaps are important

Irrespective of the above, structural coverage assessment is necessary!

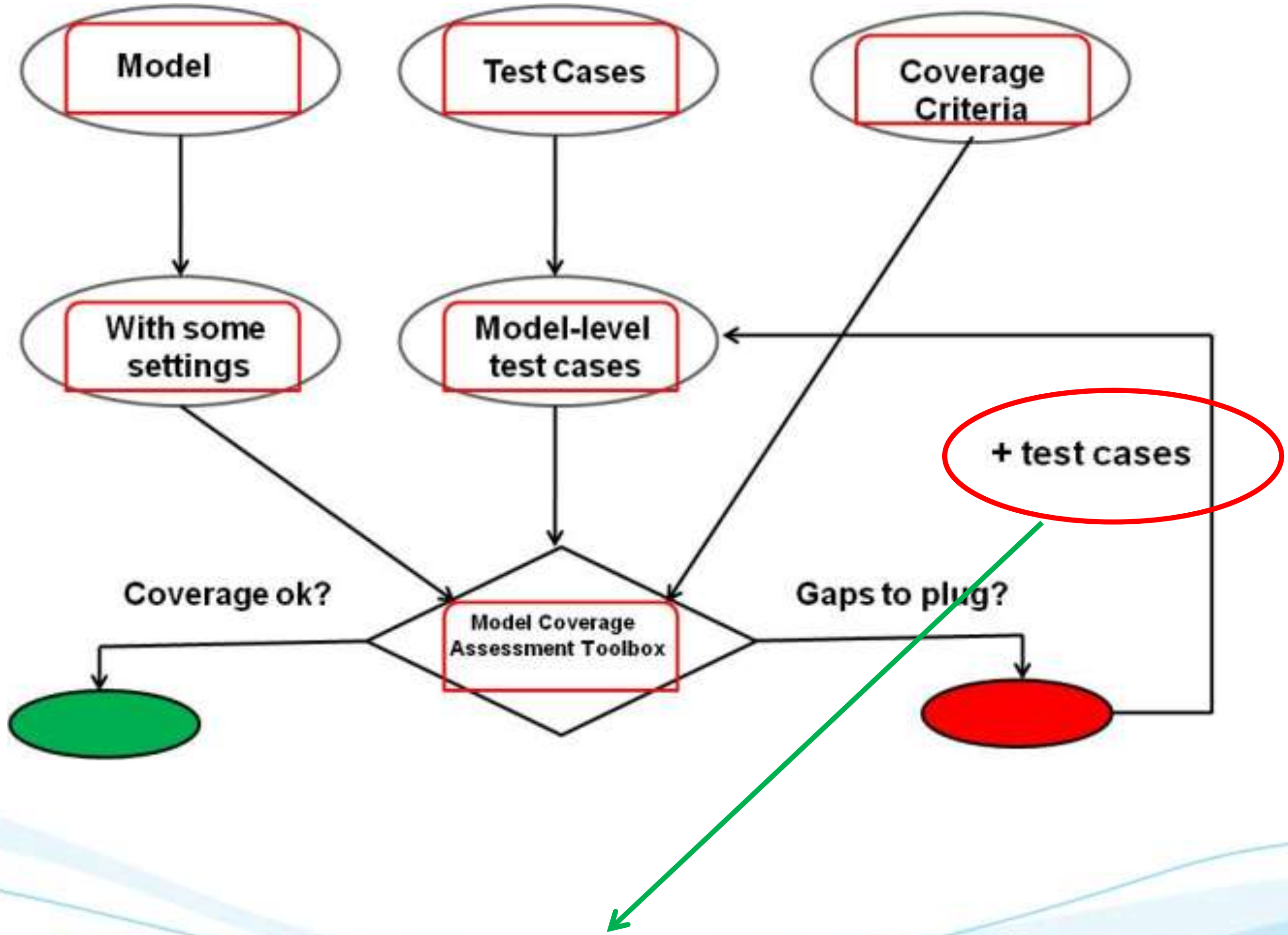
Improvements can only happen after assessment!

Simulink Design Verifier (SDV) toolbox

- SDV – Automatic Test Generation (ATG)
  - The toolbox can generate test cases automatically as per user-defined coverage requirements

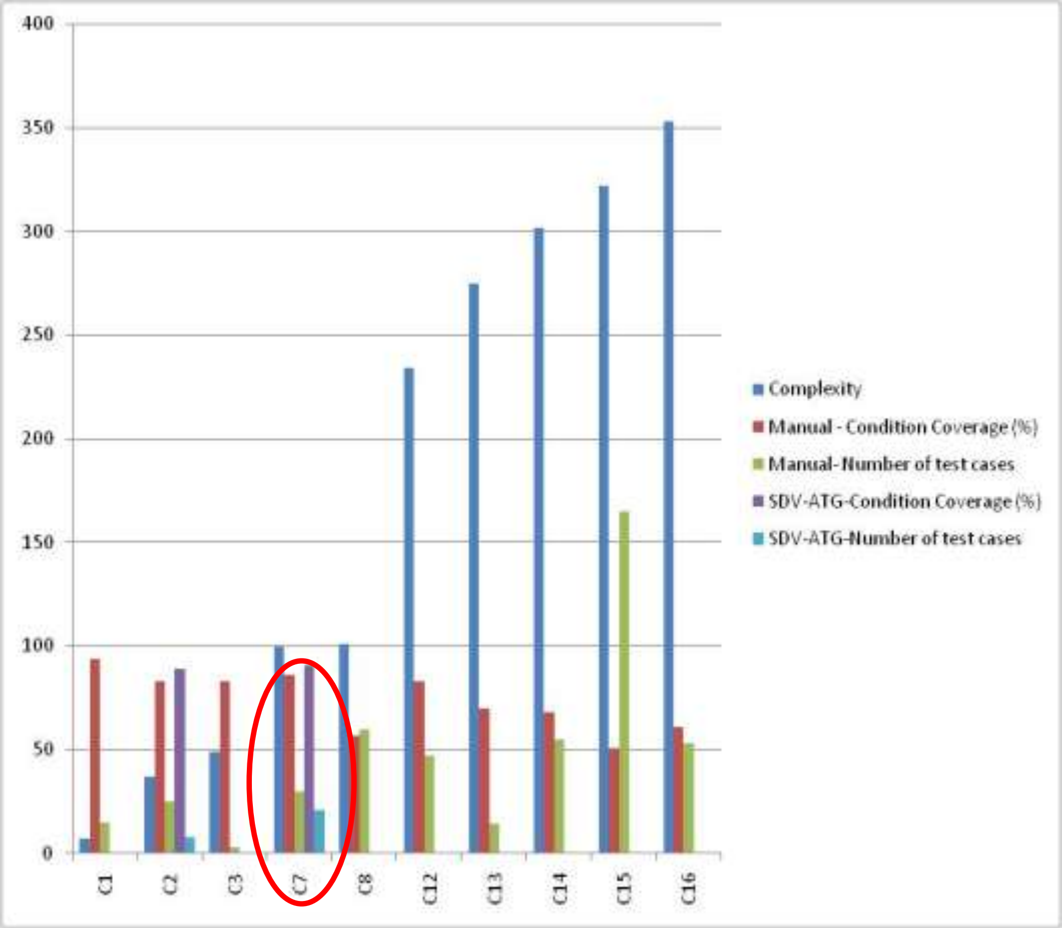
- SDV – Property Proving (PP)
  - A technique to check if the model satisfies critical requirements without writing numerous test cases

SDV - ATG



Use Simulink Design Verifier for Automatic Test case Generation!!

SDV - ATG



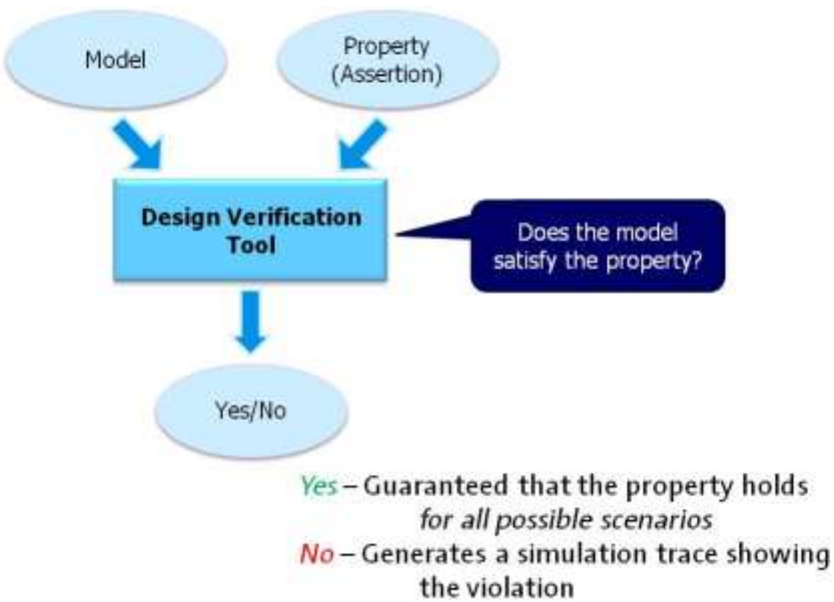
Use Simulink Design Verifier ATG capability to improve test cases further

Some points to note

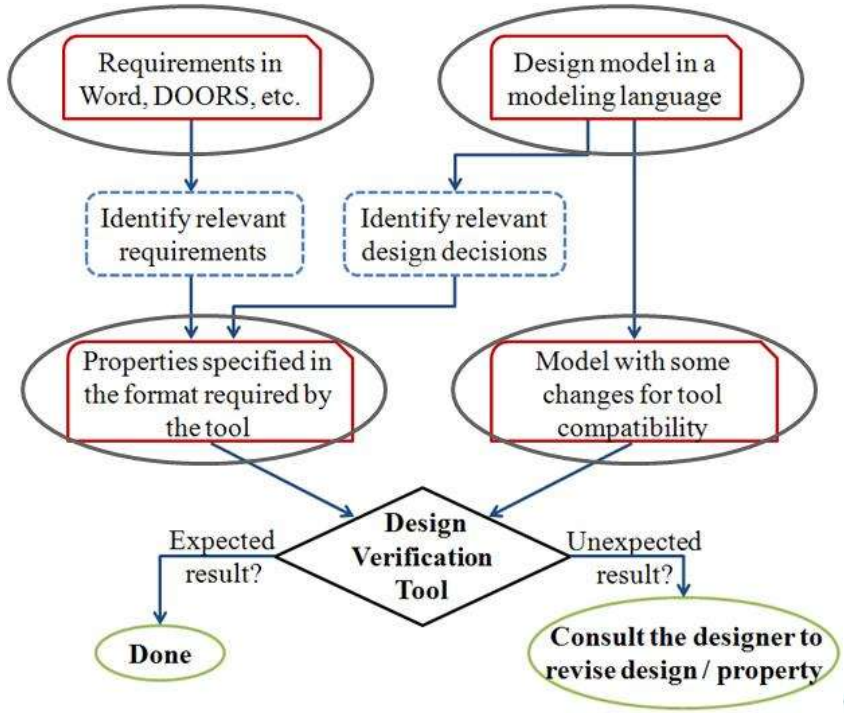
- ATG test cases to supplement existing test cases
  - First assess coverage of existing test cases
  - Identify gaps to increase coverage via self-designed test cases if desired
  - Use SDV ATG for even further improvements
- Existing models
  - May have unsupported constructs; Use automatic stubbing
  - May encounter some scalability issues
- Use ATG for selective models/subsystems
  - Where complexity is involved
  - To find out if any parts of the model are unreachable

Design Verification

Principle



Practice for Production



Relevant Mathworks toolbox:

Simulink Design Verifier (SDV)

Some Example Properties for Proving

Aero Shutter is

***never* closed if**

the speed is less than 50 kmph.

Always, if the Aero Shutter is closed,
it implies that
the coolant temperature is less than some defined maximum (92 degC).

Once ON, heater coolant pump should **remain ON**
for at least 30s
even if
the request becomes FALSE in the meantime.

Demos

Indicate some workflows

for V&V and SDV toolboxes

through short demos

Final Conclusions

- Structural coverage assessment using the V&V toolbox important to improve on test cases
- Standards recommend it - not just for critical applications
- Workflows could be tailored and adopted to suit particular production environments
- SDV toolbox capabilities could be used to improve test cases via ATG for uncovered objectives
- In addition, Property Proving feature of the SDV toolbox complements traditional testing approaches to increase overall confidence