Developing Train Propulsion Controls using automatic Model Generation and automated Build and Test

Han Geerligs
System Designer
Vehicle Engineering
Strukton Rolling Stock
19 June 2018

Strukton Rolling Stock

- Strukton Rolling Stock develops traction converters from 100 kW up to 6 MW
- Vehicle Engineering develops control software for traction converters
- The application development team consists of approximately 10 engineers
- Embedded software is developed using Model Based Design and code generation
- Using MBD since 2000, developing Embedded Targets for Analog Devices DSP

Project all over the world!

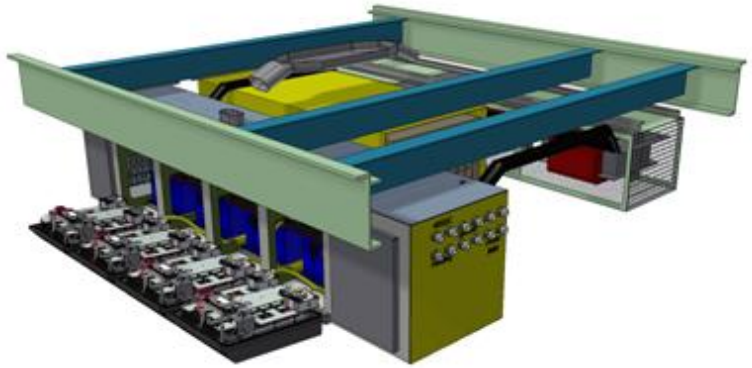
Strukton Rolling Stock

Driverless vehicle (France)

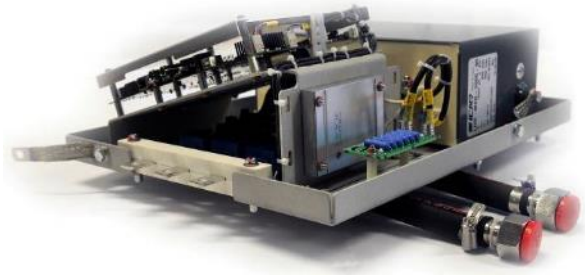
5 MW traction (India)

Control board including DSP and FPGA

Hybrid EMU (UK)



Monorail (Kuala Lumpur)



Power Module driver board

Strukton

Key Takeaways

1. Improve development using generated network interface models
2. Detect errors as they are introduced by automatic unit testing
3. Enable concurrent engineering by automated builds

Innovation Challenges and Achievements

CHALLENGES

- Developing interfaces repetitive, error prone and shared
- Build process slow and not repeatable
- Regression

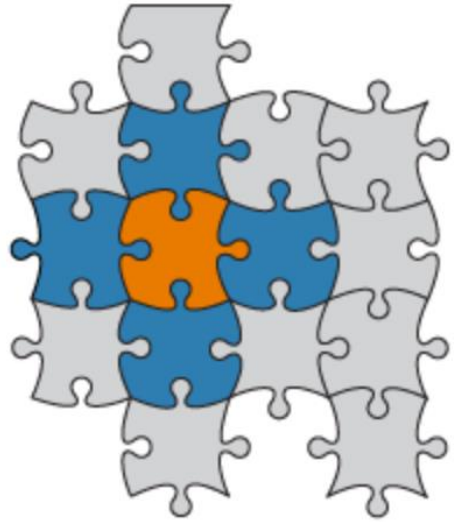
ACHIEVEMENTS

- Interface models generated automatically
- The build process is scalable
- Prompt feedback on errors

How did we get there and leverage MathWorks

- Using the Simulink API to generate interface models
- MATLAB Unit test framework
- Use MATLAB command line interface
- Benefits include:
  - Decreasing application build time (by 50%)
  - All defined by code
  - Build and test scripts start running automatically

```
add_block('simulink/Math Operations/Gain','vdp/Five','Gain','5')
```

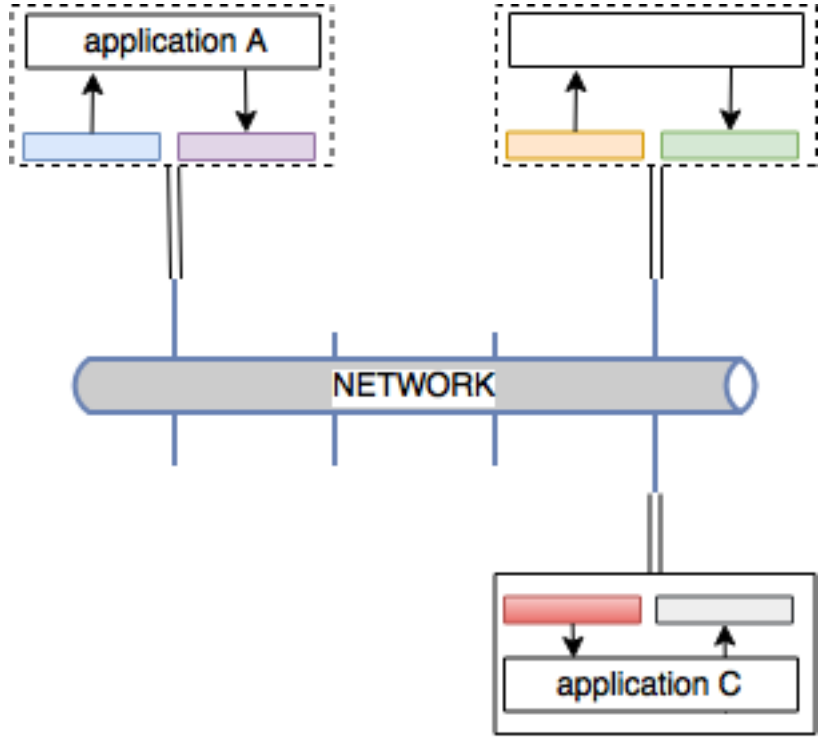


```
16:00:22 |======================== Starting Matlab ========================
16:00:22 || Version:  R2018a
16:00:22 || Location: C:\Program Files\MATLAB\R2018a\bin\matlab.exe
16:00:22 || Command: addmodelpath; ;;try exit(autobuild(%BUILD_NUMBER%, 'Releasestructure', 1)); catch exit(1); end;
16:00:22 || Max execution time: 120 minute(s)
16:00:22 || Use32bit: null
16:00:22 |================================================================
16:00:22
```

Generating interface models

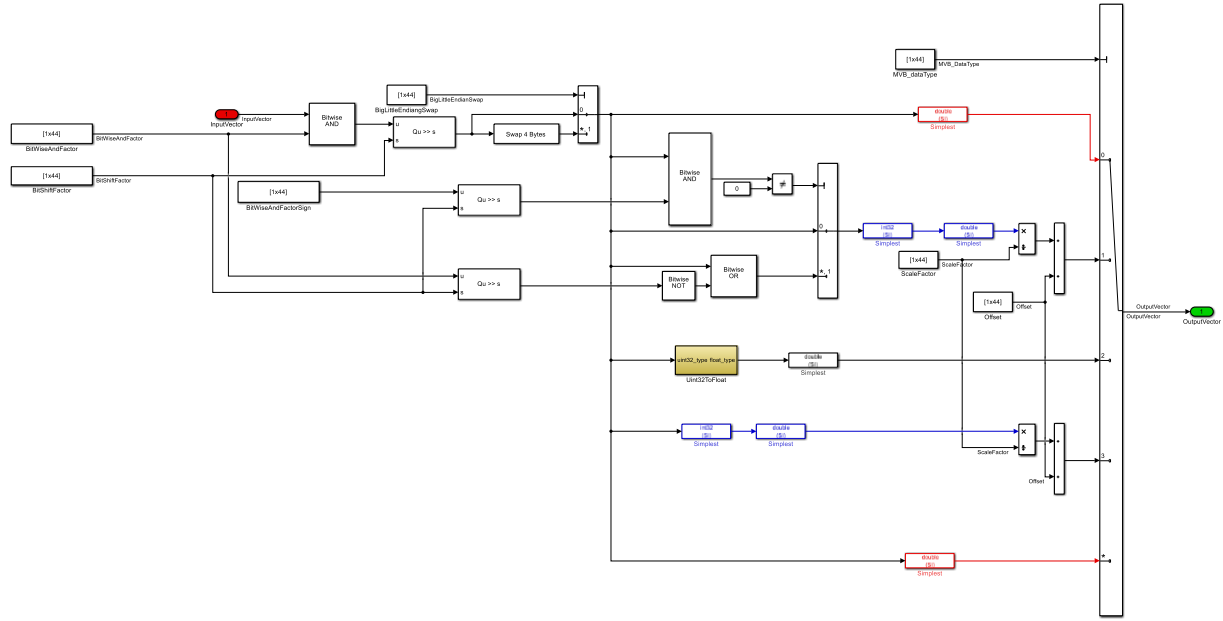
- Multiple network types (e.g. CAN, MVB, Ethernet)
- Using interface definition in Excel

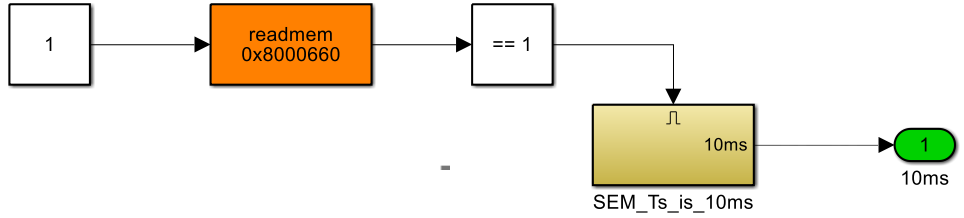
	A	B	C	D	E	F	G	H	O
1	VCMS_CAN								
2									
3	Signalname	Data type	Size in use [bits]	Byte	Bit	Byte.bitnr.	Min. Value	Max. Value	Description
630	_spare	byte	8	77	0	77.0			
631	_spare	byte	8	78	0	78.0			
632	_spare	byte	8	79	0	79.0			
633	Ctr_Forced_SetTorque_Axle_1 MSB	byte	2	80	0	80.0			Torque set point value in forcing mode for Axle 1 MSB
634	Ctr_Forced_Mode_Axle_1	bool	1	80	2	80.2			Forcing mode on Axle 1 enabling signal
635	Ctr_Forced_Torque_Dir_Axle_1	bool	1	80	3	80.3			Forcing direction on Axle 1 signal
636	Ctr_MC_Open_Request_Axle_1	bool	1	80	4	80.4			Open MC Request on Axle 1
637	Ctr_LC_Open_Request_Axle_1	bool	1	80	5	80.5			Open LC Request on Axle 1



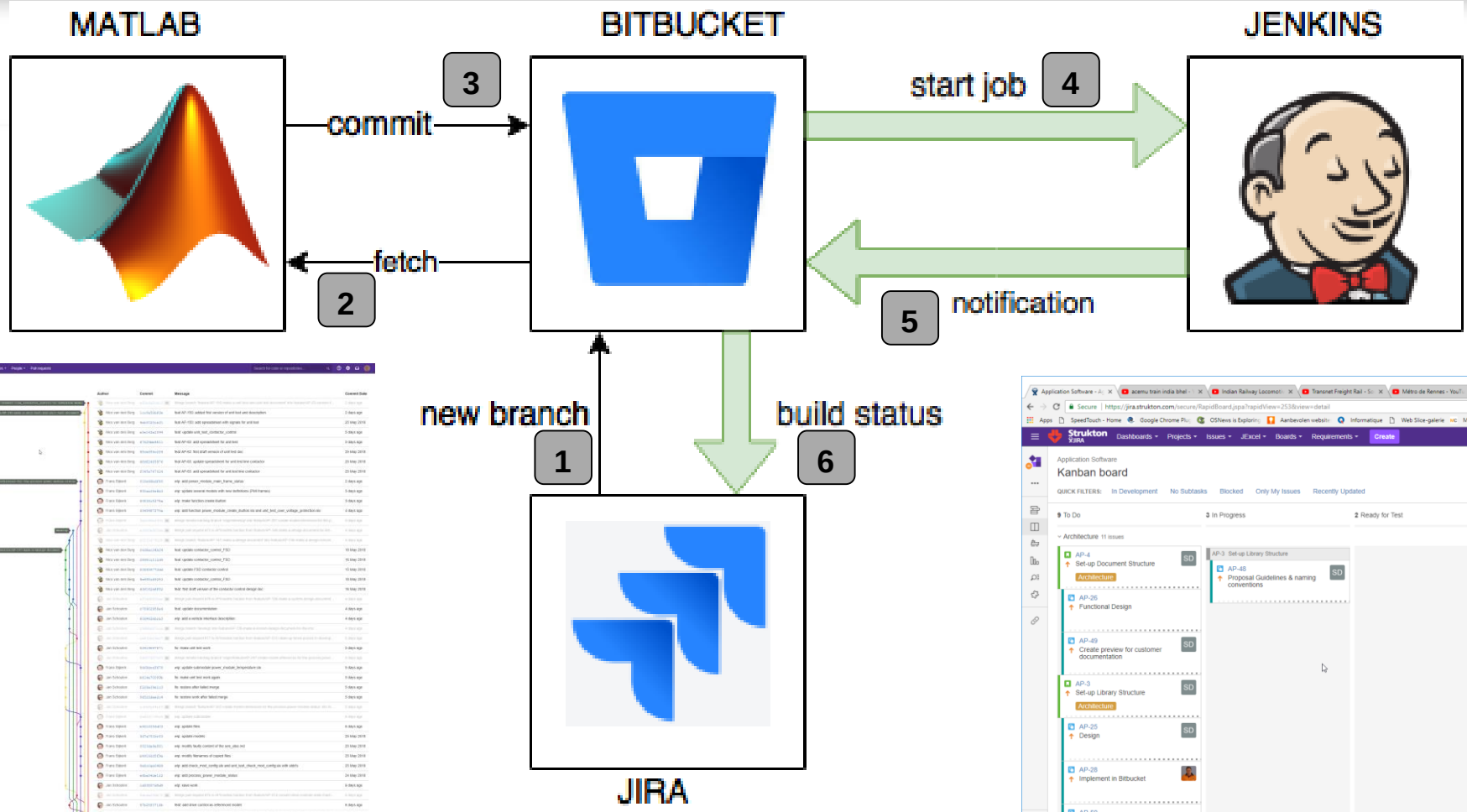
Interface model options

- Packing and unpacking
- Timing semaphores
- Multiplexed messages
- Configurable output: ports or data stores
- Configurable naming conventions
- Configurable test points and fixations
- Optimise models to enable optimisation





IT landscape and workflows



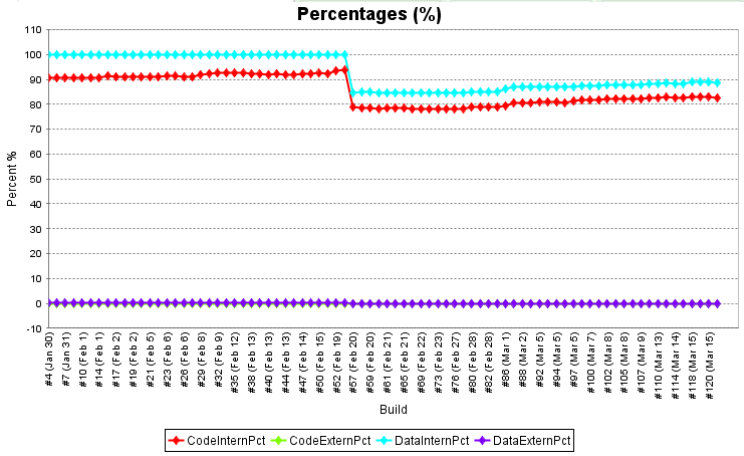
Automate the build process

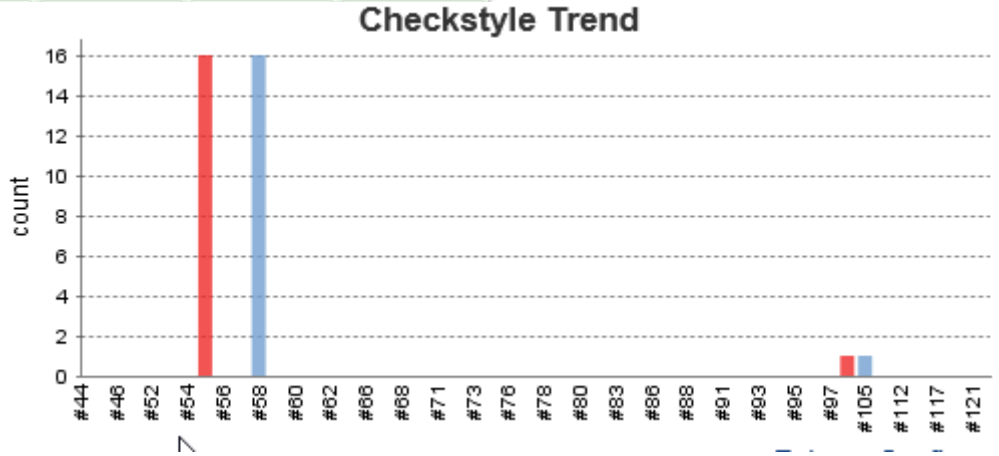
- Technology stack: Jenkins, Bitbucket, Jira and MATLAB/Simulink
- Web technology
- Build process is defined for Jenkins and MATLAB, stored in BitBucket
- Using Git and web-applications enables concurrent distributed developing

Build server

Stage View

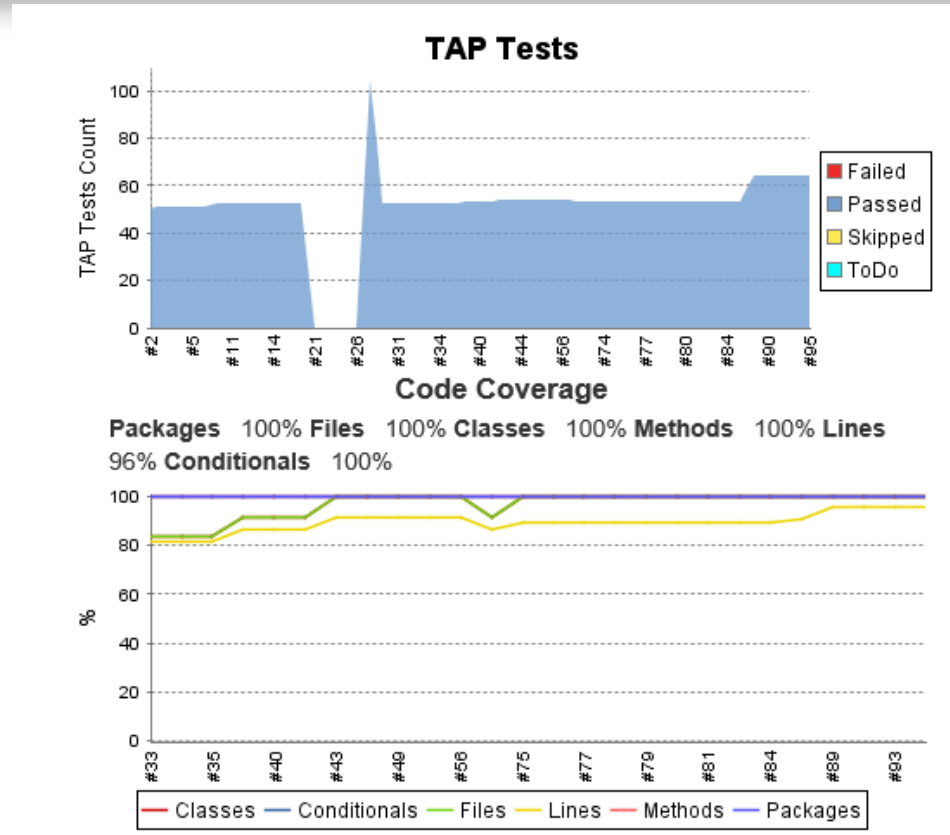
	Checkout	Cleanup	Build	Postbuild	CheckStyle	Plot	compare memory thresholds
Average stage times: (Average full run time: ~34min 58s)	10min 14s	114ms	21min 44s	3s	2s	496ms	3s
#160 Jun 11 08:22 No Changes	9min 27s	86ms	18min 33s	2s	5s	384ms	4s
#159 Jun 06 14:21 No Changes	9min 19s	96ms	32min 13s	3s	255ms	577ms	3s
#158 Jun 01 10:08 No Changes	10min 21s	110ms	32min 22s	8s	6s	740ms	5s
#157 Jun 01 08:23 No Changes	9min 48s	130ms	6min 29s failed				





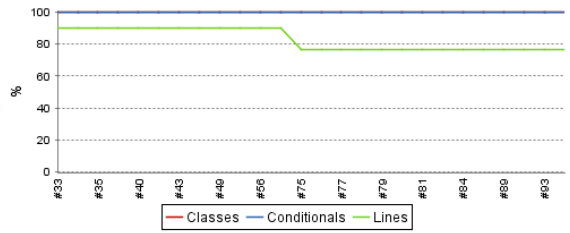
Automate the unit tests

- Cobertura and TAP output capabilities
- Define tests in MATLAB, stored in BitBucket and run on Jenkins
- ‘Codifying’ enables quick recreation of test environment



Code coverage report

Trend



File Coverage summary

Name	Classes	Lines	Conditionals
@Track/CompleteSegmentConstant.m	100% 1/1	77% 33/43	100% 0/0

Coverage Breakdown by Class

Name	Lines	Conditionals
Track\CompleteSegmentConstant	77% 33/43	N/A

Source

@Track/CompleteSegmentConstant.m

```
1      function CompleteSegmentConstant(obj, segmentNumber)
2      %COMPLETECONSTANTSEGMENT Summary of this function goes here
3      %   Detailed explanation goes here
4
5   53     obj.TraceLog.startsection(['in function ' mfilename]);
6
7          %% Constant Force
8   53     trainSpeed =  min(min(obj.SegmentVmaxMatrix(segmentNumber,:)), obj.route.Segment(segmentNumber-1).SpeedResult(end));
9   53     [Fturn, Fangle, FresAir, FresRol, FresTrain] = CalculateForce(obj, trainSpeed, segmentNumber);
10  53     Fconstant = (FresAir + FresRol + FresTrain + Fangle + Fturn)*obj.EnableResistiveForce;
11
12         %% Calculate distance, speed and time
13  53     segmentDistance = obj.route.Segment(segmentNumber).Distance - ...
14  53         obj.route.Segment(segmentNumber-1).DistanceResult(end);
15
16  53     if obj.route.Segment(segmentNumber-1).SpeedResult(end) > 0
17  53         tSegment = (segmentDistance/(min(min(obj.SegmentVmaxMatrix(segmentNumber,:)), obj.route.Segment(segmentNumber-1).SpeedResult(end))/3.6));
18   0     else
19   0         tSegment = (segmentDistance/(min(obj.SegmentVmaxMatrix(segmentNumber,:))/3.6));
20         end
21         %% Resize if segmenttime >= 1 second
```

Concluding Remarks

Our best practices, learnings and recommendations

- Consolidate knowledge, workflows and infrastructure by ’codifying’
- Use BitBucket to deploy build features automatically, e.g. display of code and data size
- Connect to standard, web-based applications
- Prefer "push" over "pull“
- Helps improving quality (non-regression)

Forward looking plans

- Generate documentation from models and publish to Confluence (already started)
- Add model coverage and guideline checking to the automated V&V activities
- Deploy applications to test setups automatically
- Explore Simulink Test and Simulink Check