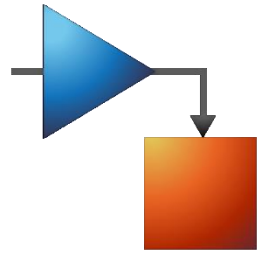
Toolchain Definition and Integration for ISO 26262-Compliant Development

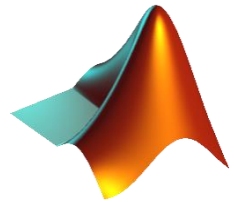
Dave Hoadley, PhD
June 2020

Introduction

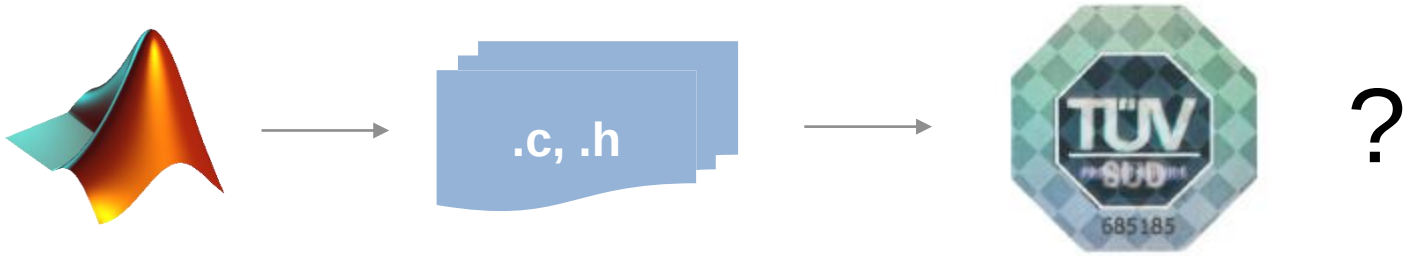
- MathWorks tools like Simulink and Stateflow are established as suitable for generating code for ISO 26262 QM to ASIL-D applications



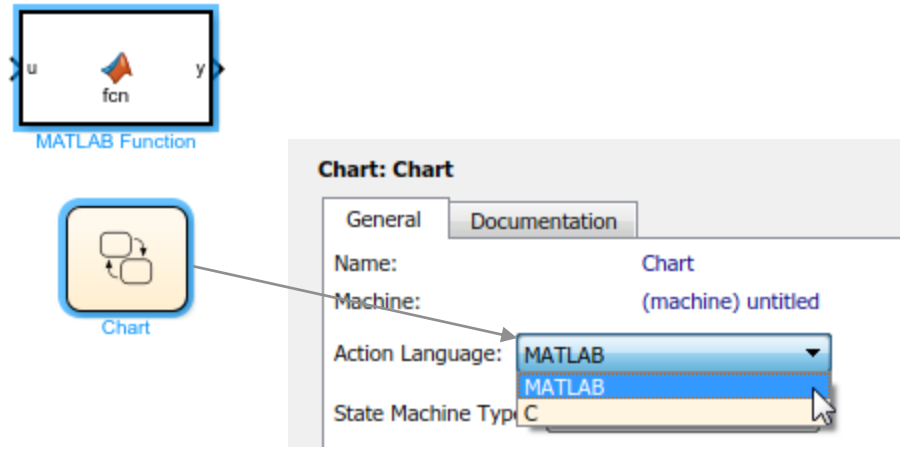
- MATLAB has emerged for AD/ADAS algorithm prototyping
  - A natural language for matrices, image processing, deep learning
  - MATLAB source (text) is also seamless to integrate with Agile workflow tools



- Can we generate certifiable code from MATLAB?



Yes! MATLAB and Simulink Integration

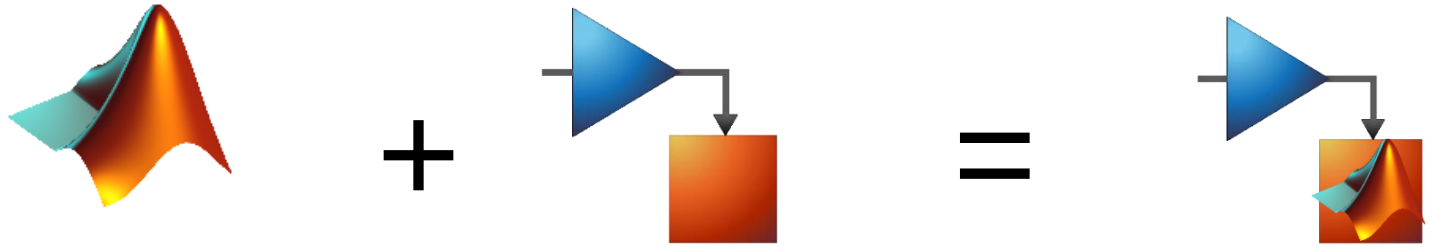


- Called by the MATLAB Function block and/or Stateflow
  - Inlined MATLAB operators
  - External functions
  - Long list of language features that support code generation
  - And functions, including toolboxes like Sensor Fusion, Stats and Machine Learning, Automated Driving, Deep Learning

- MATLAB code generation is supported by our IEC Certification Kit and (Simulink) reference workflow

Algorithm Designer Win-win

- We can combine these and have the best of both worlds
  - + Richness of the MATLAB language
  - + Rigor of the Simulink family of verification tools



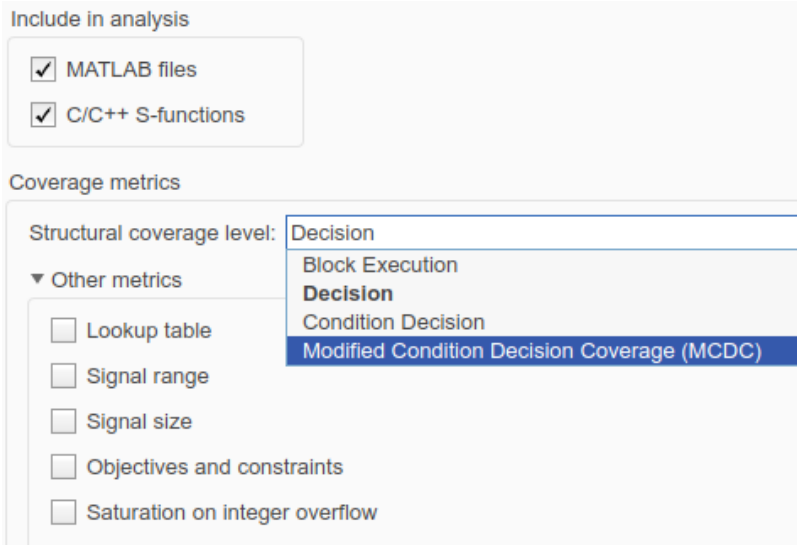
- "I'm a MATLAB user, is Simulink for me?"

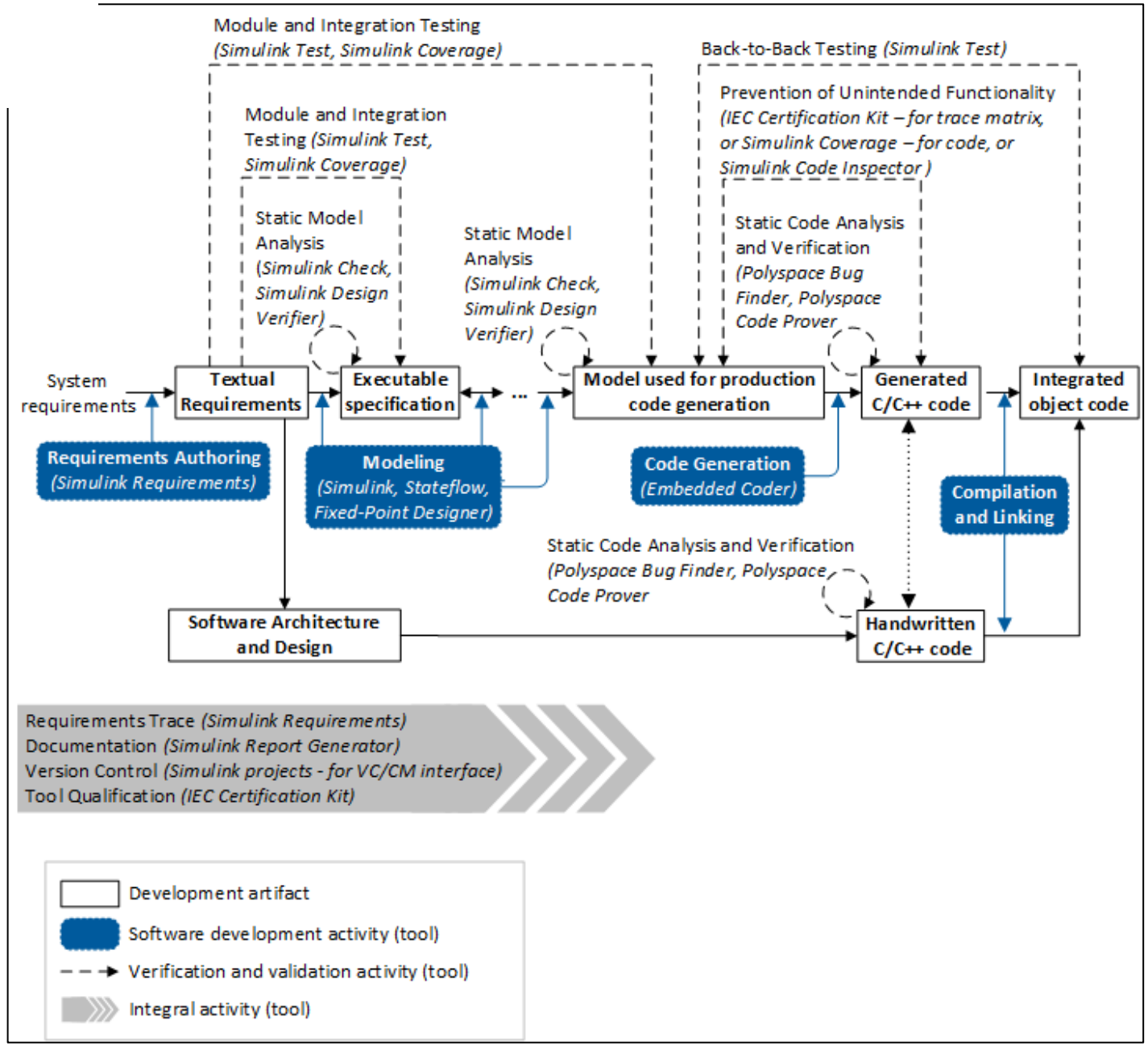
➔ If you need to provide **evidence of conformance**

➔ To define **architecture** around MATLAB algorithms

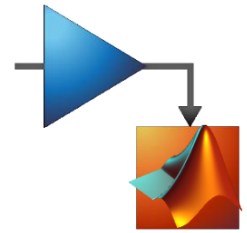
Verification workflow

- Trace requirements ⇔ design ⇔ implementation ⇔ validation
- Meet design & implementation standards
- Show intended and no unintended functionality
  - Coverage is key evidence





MATLAB + Simulink ISO 26262 Workflow



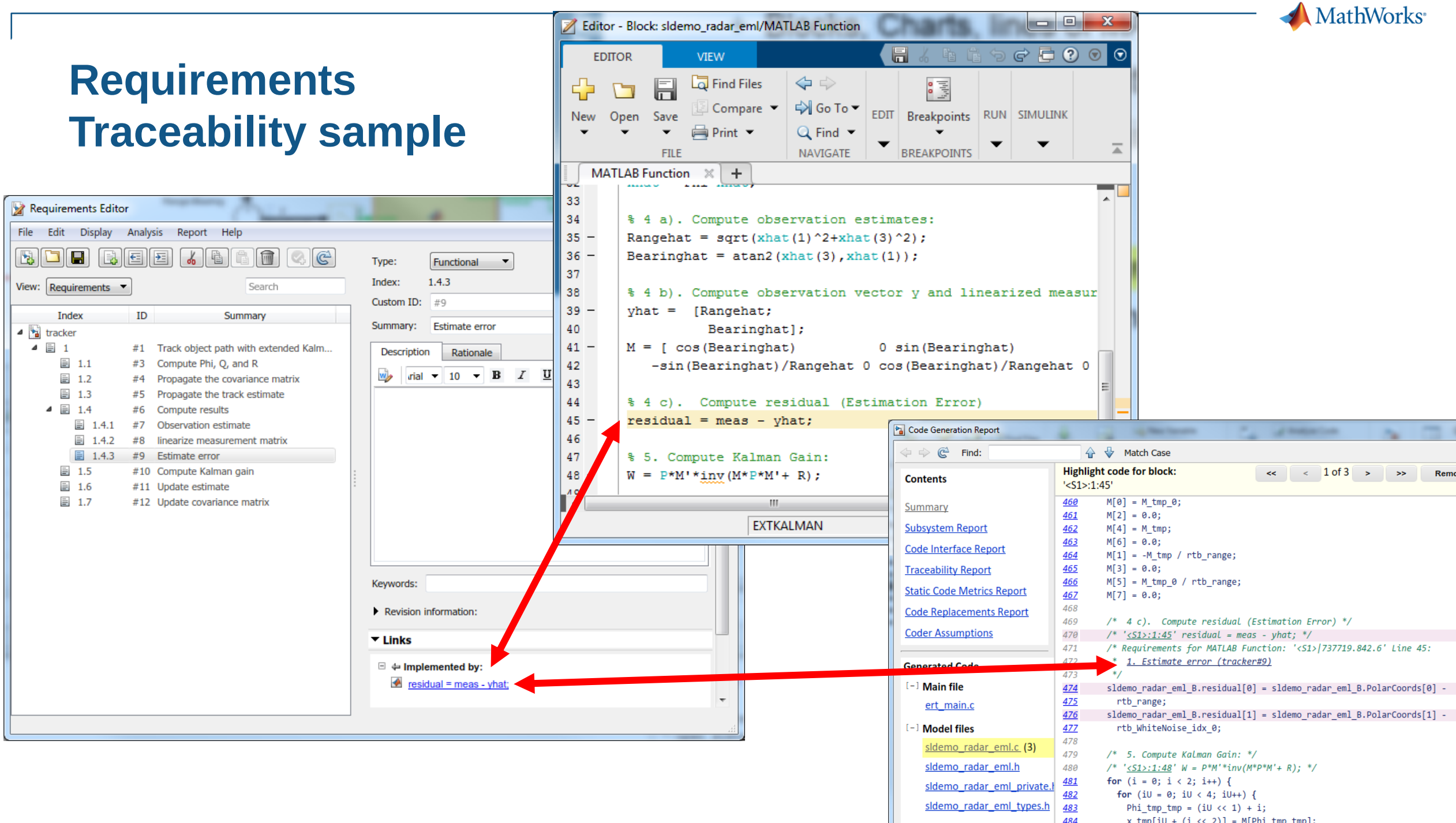
- Our reference workflow supports this combined language
  - + Requirements traceability
  - + Design standards
  - + Prove correct functionality
  - + Prove absence of unintended functionality

Traceability

Simulink Requirements

- Simulink Requirements supports authoring, importing/exporting, and linking requirements to model elements, test cases (Simulink Test)
  - Blocks, Charts, **lines of MATLAB code**
- Requirements Traceability report for evidence
- **MATLAB source** and user comments can be included as generated comments

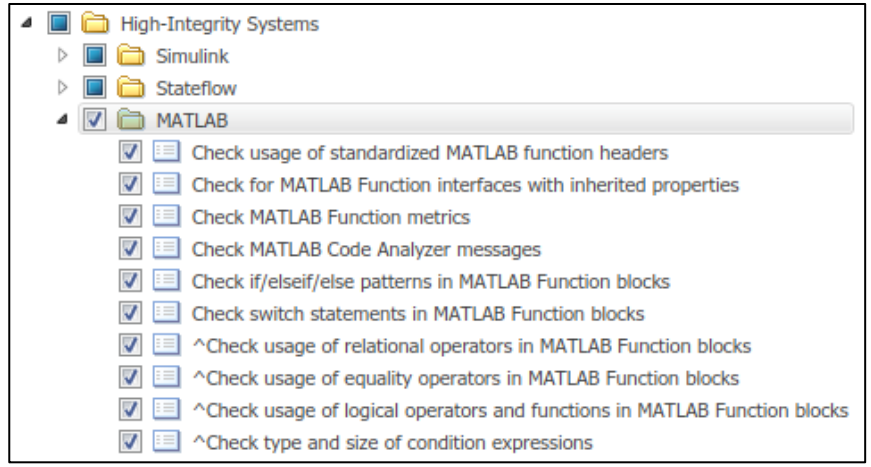
Requirements Traceability sample



Design and Code Standards

Simulink Check

- Simulink Check has checks for MATLAB style and improving code compliance
  - Enforcement of low complexity
  - Enforcement of comment density
  - Strong data typing between MATLAB and Simulink
  - Find logical operators with mixed data types
- Some MATLAB & Embedded Coder settings for MISRA-C
- MATLAB guidelines are emerging (JMAAB)



- More MATLAB checks are needed

Demonstrate correct functionality

Simulink Requirements

Simulink Test

Simulink Design Verifier

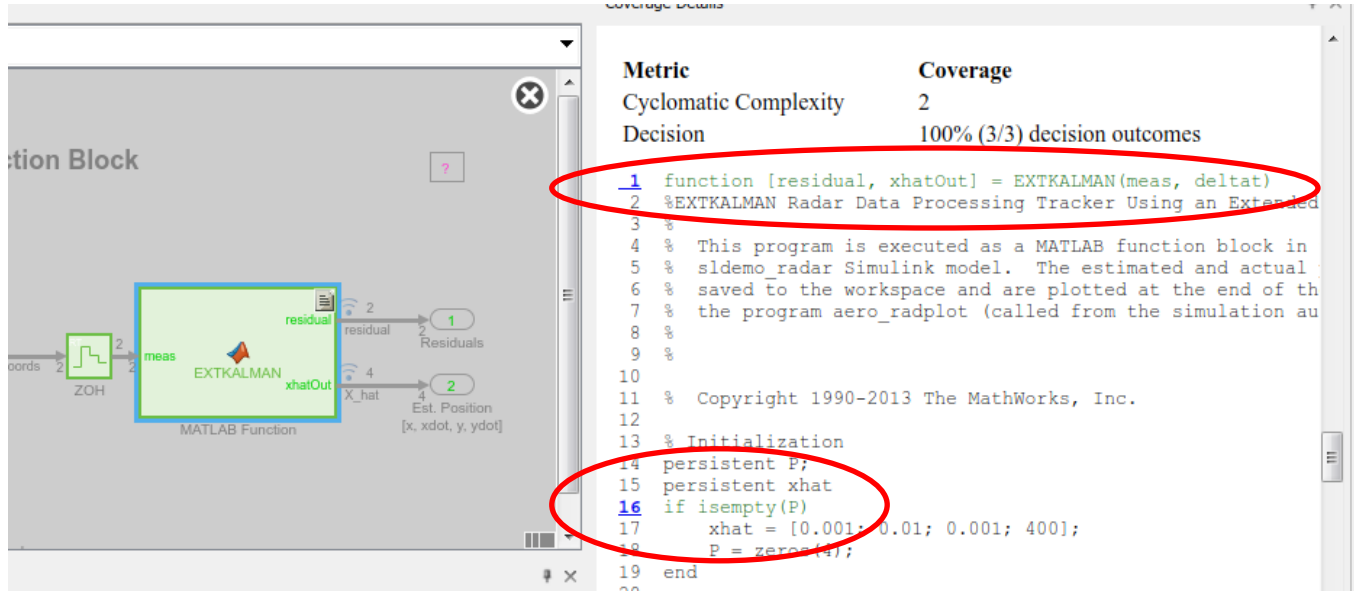
- Requirements-based test authoring, execution via Simulink Test
- Simulink Design Verifier (SLDV) property proving
- SLDV design error detection
- Back to back testing for model to code for Software-in-the-Loop (SIL), Processor-in-the-Loop (PIL)

Demonstrate no unintended functionality

Simulink Coverage

Simulink Design Verifier

- Simulink Coverage to show completeness of test cases
  - Model coverage
  - Code coverage for SIL/PIL
- SLDV can generate missing tests

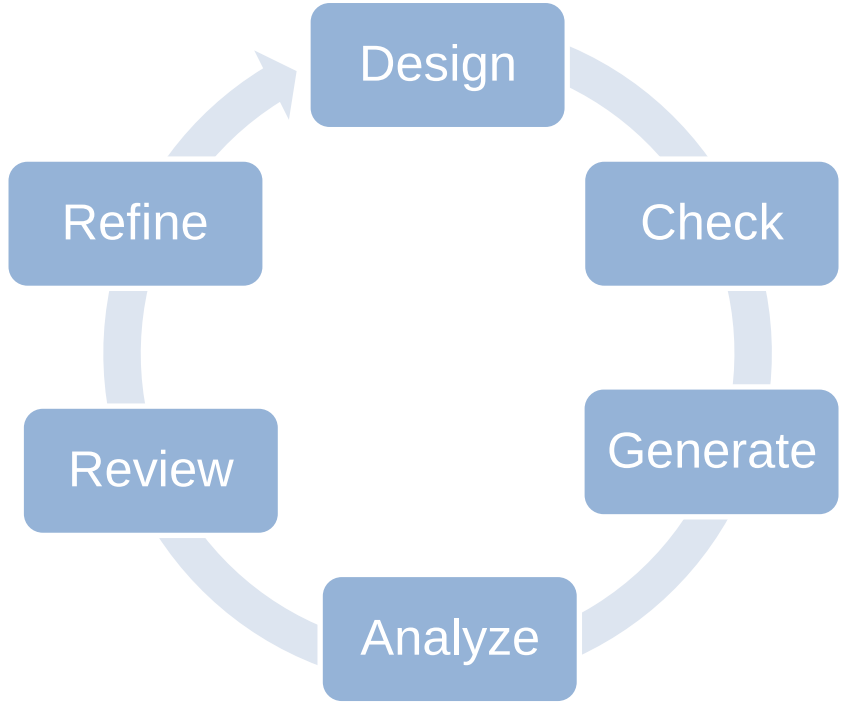


Summary so far

- Customers are successfully using MATLAB in ISO 26262-compliant products today
- Our verification workflow and tools support MATLAB called by Simulink
- But… there are some gaps remaining
  - Potential issues with MISRA-C compliance of code generated from MATLAB
  - Achieving MATLAB vs C code coverage
  - Simplifying Simulink model ~~comparison~~ reviews

Code standards compliance

- Practice is to
  - run model checks — **Simulink Check**
  - generate code
  - analyze compliance — **Polyspace Bug Finder**
- Issues discovered?
  - document and proceed
  - rework the algorithm
  - rewrite a compliant function (toolboxes)
- Result is an allowed function list (*language subset*)
- Process gets more efficient over time



Code coverage

- MATLAB functions can be complex in C/C++

```
% 5. Compute Kalman Gain:
W = P*M'*inv(M*P*M'+ R);
```

- One test case gets coverage in MATLAB, but more required to show no unintended functionality in the generated C

- Strategies include
  - Develop unit tests for feature/function
  - Implement a simpler replacement

```
480     /*  5. Compute Kalman Gain: */
481     /* '<S1>:1:48' W = P*M'*inv(M*P*M'+ R); */
482     for (i = 0; i < 2; i++) {
483       for (iU = 0; iU < 4; iU++) {
484         Phi_tmp_tmp = (int32_T)((int32_T)(iU << 1) + i);
485         x_tmp[(int32_T)(iU + (int32_T)(i << 2))] = M[Phi_tmp_tmp];
486         M_0[Phi_tmp_tmp] = 0.0;
487         Phi_tmp = (int32_T)(iU << 2);
488         M_0[Phi_tmp_tmp] += sldemo_radar_eml_DWork.P[Phi_tmp] * M[i];
489         M_0[Phi_tmp_tmp] += sldemo_radar_eml_DWork.P[(int32_T)(Phi_tmp + 1)] *
490           0.0;
491         M_0[Phi_tmp_tmp] += sldemo_radar_eml_DWork.P[(int32_T)(Phi_tmp + 2)] *
492           M[(int32_T)(i + 4)];
493         M_0[Phi_tmp_tmp] += sldemo_radar_eml_DWork.P[(int32_T)(Phi_tmp + 3)] *
494           0.0;
495       }
496     }
497
498     for (i = 0; i < 2; i++) {
499       for (iU = 0; iU < 2; iU++) {
500         Phi_tmp_tmp = (int32_T)(i << 2);
501         Phi_tmp = (int32_T)((int32_T)(i << 1) + iU);
502         Phi_1[Phi_tmp] = (((x_tmp[(int32_T)(Phi_tmp_tmp + 1)] * M_0[(int32_T)(iU
503           + 2)] + x_tmp[Phi_tmp_tmp] * M_0[iU]) + x_tmp[(int32_T)(Phi_tmp_tmp +
504           2)] * M_0[(int32_T)(iU + 4)]) + x_tmp[(int32_T)(Phi_tmp_tmp + 3)] *
505                       M_0[(int32_T)(iU + 6)]) + R[Phi_tmp];
506       }
507     }
508
509     if (fabs(Phi_1[1]) > fabs(Phi_1[0])) {
510       rtb_range = Phi_1[0] / Phi_1[1];
511       rtb_WhiteNoise_idx_0 = 1.0 / (rtb_range * Phi_1[3] - Phi_1[2]);
512       M_tmp = Phi_1[3] / Phi_1[1] * rtb_WhiteNoise_idx_0;
513       M_tmp_0 = -rtb_WhiteNoise_idx_0;
514       y_idx_2 = -Phi_1[2] / Phi_1[1] * rtb_WhiteNoise_idx_0;
515       rtb_WhiteNoise_idx_0 *= rtb_range;
516     } else {
517       rtb_range = Phi_1[1] / Phi_1[0];
518       rtb_WhiteNoise_idx_0 = 1.0 / (Phi_1[3] - rtb_range * Phi_1[2]);
519       M_tmp = Phi_1[3] / Phi_1[0] * rtb_WhiteNoise_idx_0;
520       M_tmp_0 = -rtb_range * rtb_WhiteNoise_idx_0;
521       y_idx_2 = -Phi_1[2] / Phi_1[0] * rtb_WhiteNoise_idx_0;
522     }
```

Reviewing Simulink models

- Classic approaches
  - 1-1 or 1-many at desk or in conference rooms
  - Screen sharing apps

- Modern workforces are often distributed and busy, making this a challenge

- Tools to manage the review process, such as Gerrit Code Review, are becoming a popular approach

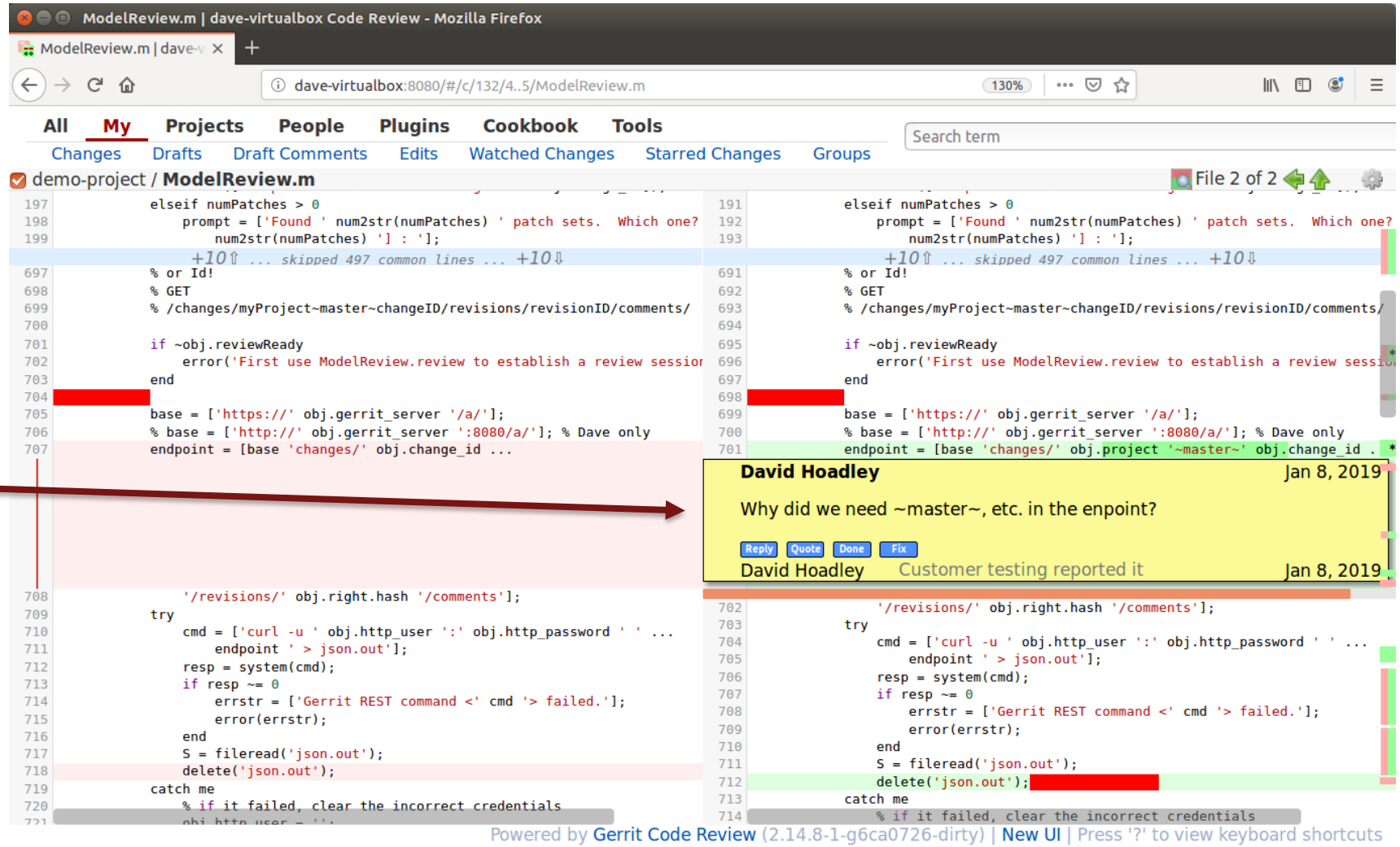
Text-based differences + review comments

Gerrit Code Review

Gerrit implements a web-based review and approval workflow for git patch revisions

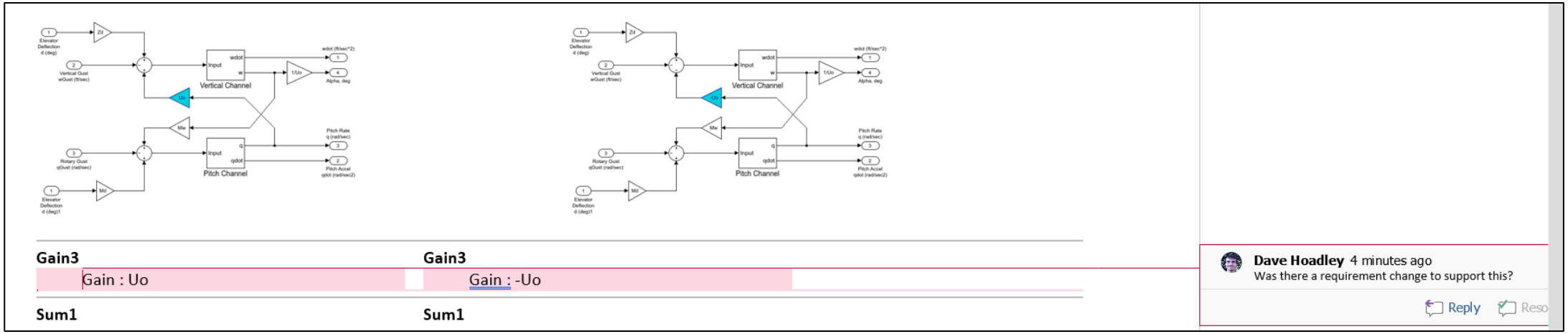
Review comments are shared **in the context** of the source

But, binary formats not supported (.slx)



Model reviews with built-in features

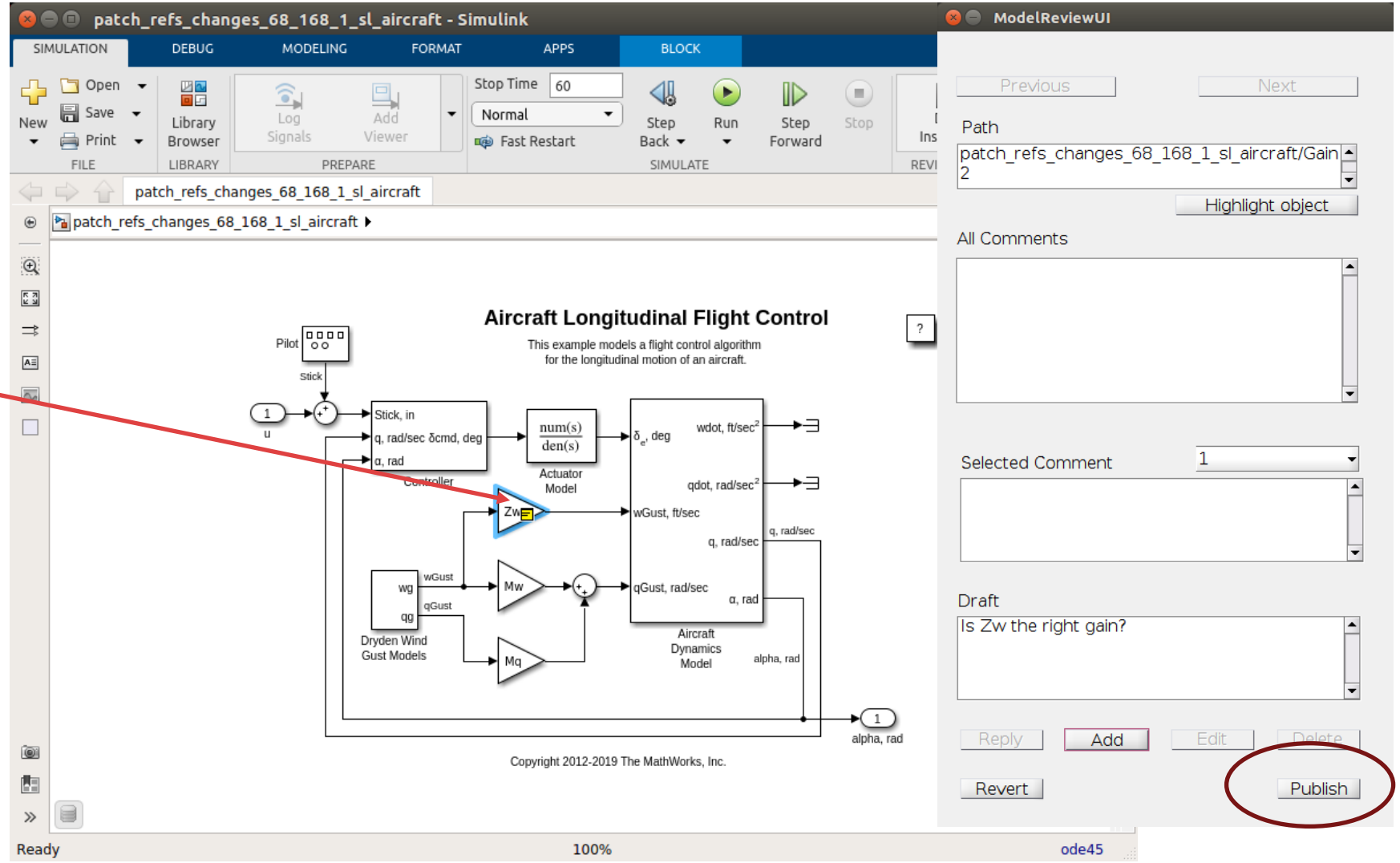
- Configure SCM with external diff tool for MATLAB files
  - E.g., "C:\Program Files\MATLAB\R2019a\bin\win64\mlDiff.exe" %LOCAL %PWD %REMOTE
  - Note this will reuse a running MATLAB not start a new instance
- Publish model comparison to MS Word format
- Annotate and share Word document with comments/replies

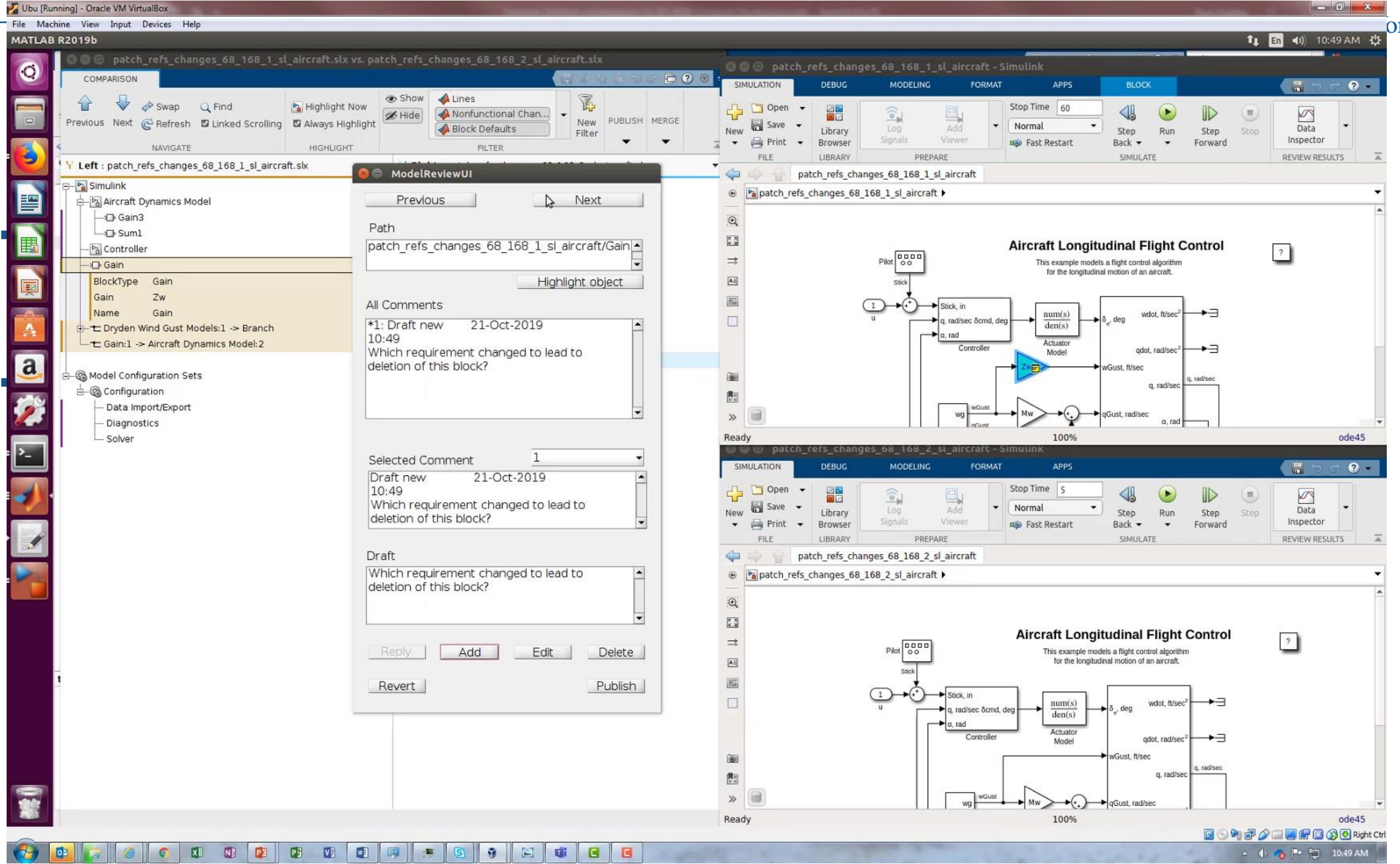


Extending this concept *into* Simulink

MathWorks Consulting Services

- Custom add-on to Simulink context menu
- Block badge indicates comment attached
- Publish to Gerrit when ready to share




Ubu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MATLAB R2019b
patch_refs_changes_68_168_1_sl_aircraft.slx vs. patch_refs_changes_68_168_2_sl_aircraft.slx
COMPARISON
Previous Next Swap Refresh Find Linked Scrolling
NAVIGATE
Highlight Now Always Highlight
HIGHLIGHT
Show Hide Lines Nonfunctional Chan... Block Defaults
FILTER
New Filter
PUBLISH
MERGE
Left : patch_refs_changes_68_168_1_sl_aircraft.slx
Simulink
Aircraft Dynamics Model
Gain3
Sum1
Controller
Gain
BlockType Gain
Gain Zw
Name Gain
Dryden Wind Gust Models:1 -> Branch
Gain:1 -> Aircraft Dynamics Model:2
Model Configuration Sets
Configuration
Data Import/Export
Diagnostics
Solver
ModelReviewUI
Previous
Next
Path
patch_refs_changes_68_168_1_sl_aircraft/Gain
Highlight object
All Comments
*1: Draft new 21-Oct-2019
10:49
Which requirement changed to lead to deletion of this block?
Selected Comment
1
Draft new 21-Oct-2019
10:49
Which requirement changed to lead to deletion of this block?
Draft
Which requirement changed to lead to deletion of this block?
Reply
Add
Edit
Delete
Revert
Publish
patch_refs_changes_68_168_1_sl_aircraft - Simulink
SIMULATION DEBUG MODELING FORMAT APPS BLOCK
New Open Save Print
FILE
Library Browser
LIBRARY
Log Signals Add Viewer
PREPARE
Stop Time 60
Normal
Fast Restart
Step Back Run Step Forward Stop
SIMULATE
Data Inspector
REVIEW RESULTS
patch_refs_changes_68_168_1_sl_aircraft
Aircraft Longitudinal Flight Control
This example models a flight control algorithm for the longitudinal motion of an aircraft.
Pilot
Stick
Stick, in
q, rad/sec δcmd, deg
α, rad
Controller
num(s)
den(s)
Actuator Model
wdot, ft/sec²
qdot, rad/sec²
wGust, ft/sec
q, rad/sec
Zw
Mw
wg
wGust
qGust, rad/sec
α, rad
Ready
100%
ode45
patch_refs_changes_68_168_2_sl_aircraft - Simulink
Stop Time 5
patch_refs_changes_68_168_2_sl_aircraft
Aircraft Longitudinal Flight Control
This example models a flight control algorithm for the longitudinal motion of an aircraft.
Ready
100%
ode45
Right Ctrl
10:49 AM

Summary redux

- Customers are successfully using Simulink **AND MATLAB** in ISO 26262-compliant products today
- There are some gaps remaining
  - Potential issues with MISRA-C compliance of code generated from MATLAB
  - Achieving MATLAB to C code coverage
  - Simplifying Simulink model reviews

MathWorks Consulting Services

- See Best practices for Simulink and MATLAB for ISO-26262 for advice

Contact info and poll questions

- How are you reviewing Simulink models today?
  1. Ad hoc
  2. Screen sharing/model discussion
  3. Reviewing reports offline (html, etc.)
  4. Simulink comparison tool
  5. 3rd party model comparison tool
  6. Other

Please contact me with questions at dhoadley@mathworks.com and let me know if you would like to have a follow-up conversation