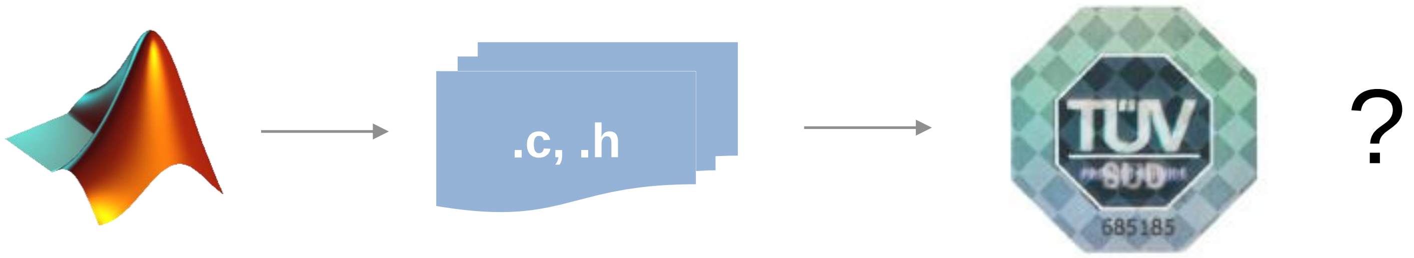
Toolchain Definition and Integration for ISO 26262-Compliant Development

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

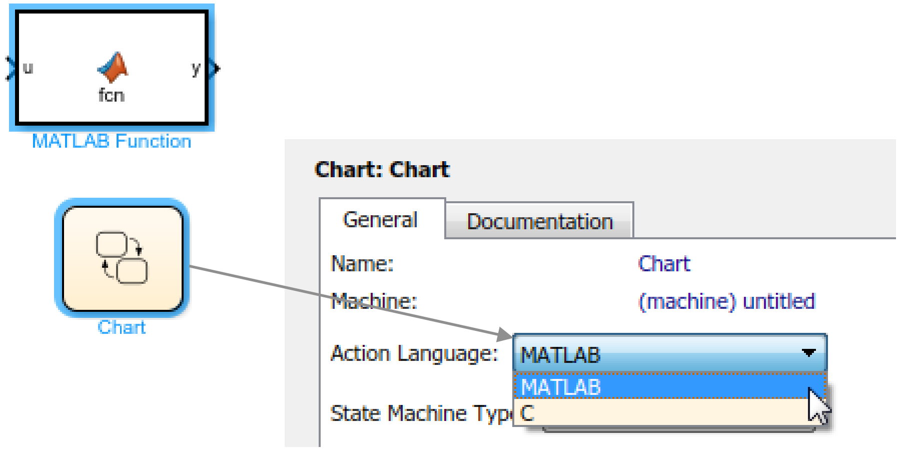


Chart: Chart

General | Documentation

Name: Chart

Machine: (machine) untitled

Action Language: MATLAB

MATLAB

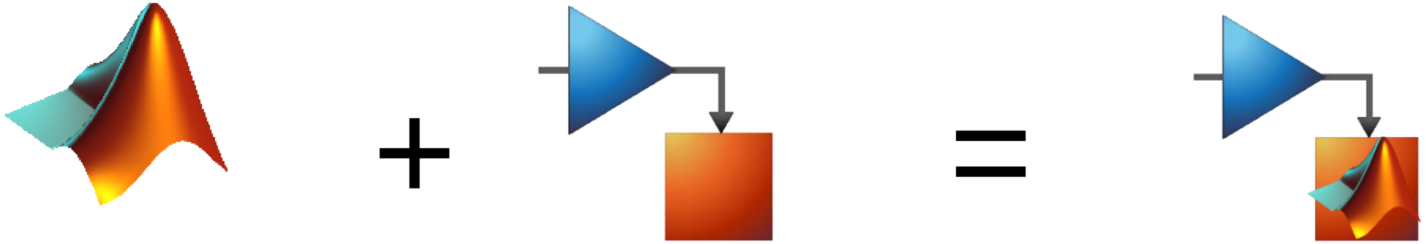
State Machine Typ C

- 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



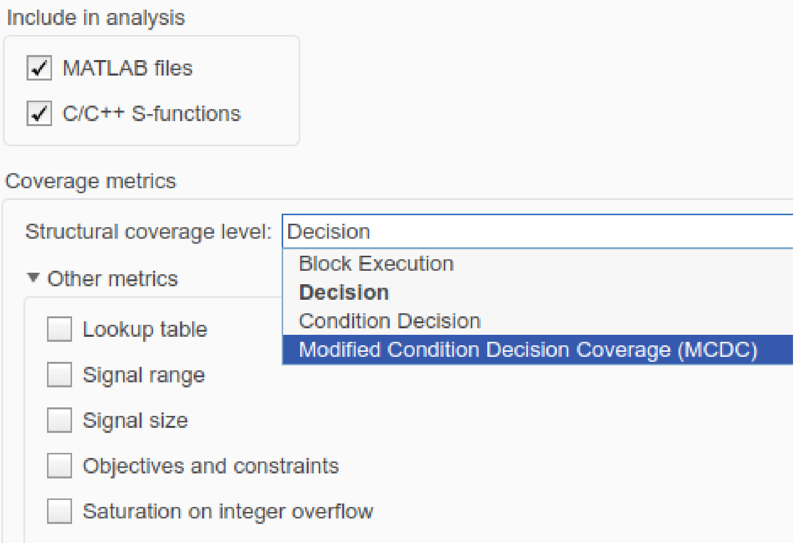
- “I’m a MATLAB user, is Simulink for me?”

➔ 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



Include in analysis

☑ MATLAB files

☑ C/C++ S-functions

Coverage metrics

Structural coverage level: Decision

Block Execution
Decision
Condition Decision
Modified Condition Decision Coverage (MCDC)

▾ Other metrics

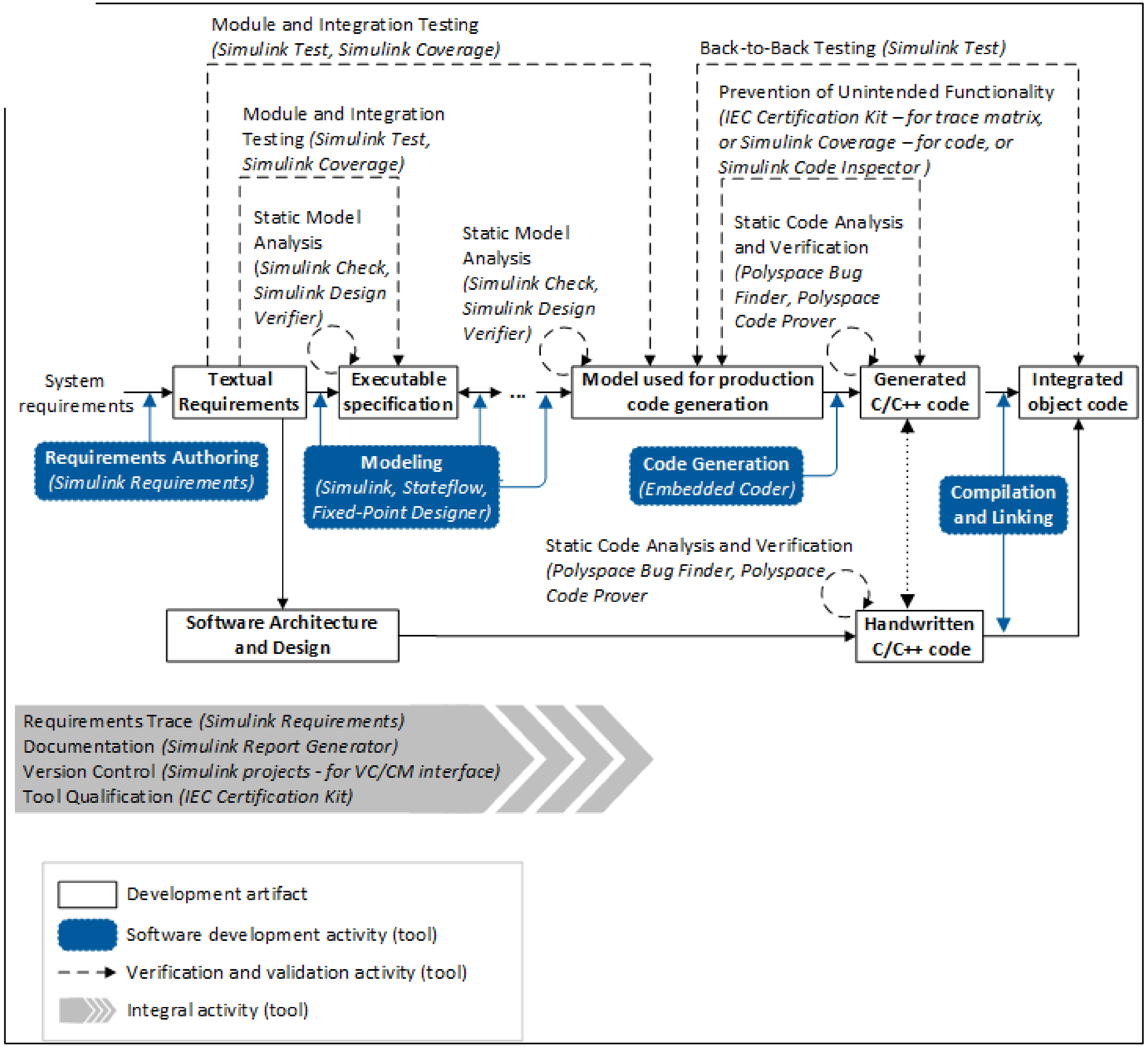
□ Lookup table

□ Signal range

□ Signal size

□ Objectives and constraints

□ Saturation on integer overflow



Module and Integration Testing *(Simulink Test, Simulink Coverage)*

Back-to-Back Testing *(Simulink Test)*

Module and Integration Testing *(Simulink Test, Simulink Coverage)*

Prevention of Unintended Functionality *(IEC Certification Kit – for trace matrix, or Simulink Coverage – for code, or Simulink Code Inspector)*

Static Model Analysis *(Simulink Check, Simulink Design Verifier)*

Static Model Analysis *(Simulink Check, Simulink Design Verifier)*

Static Code Analysis and Verification *(Polyspace Bug Finder, Polyspace Code Prover*

System requirements

Textual Requirements

Executable specification

...

Model used for production code generation

Generated C/C++ code

Integrated object code

Requirements Authoring *(Simulink Requirements)*

Modeling *(Simulink, Stateflow, Fixed-Point Designer)*

Code Generation *(Embedded Coder)*

Compilation and Linking

Static Code Analysis and Verification *(Polyspace Bug Finder, Polyspace Code Prover*

Software Architecture and Design

Handwritten C/C++ code

Requirements Trace *(Simulink Requirements)*
Documentation *(Simulink Report Generator)*
Version Control *(Simulink projects - for VC/CM interface)*
Tool Qualification *(IEC Certification Kit)*

□ Development artifact

■ Software development activity (tool)

- - → Verification and validation activity (tool)

⟫ Integral activity (tool)

MATLAB + Simulink ISO 26262 Workflow

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

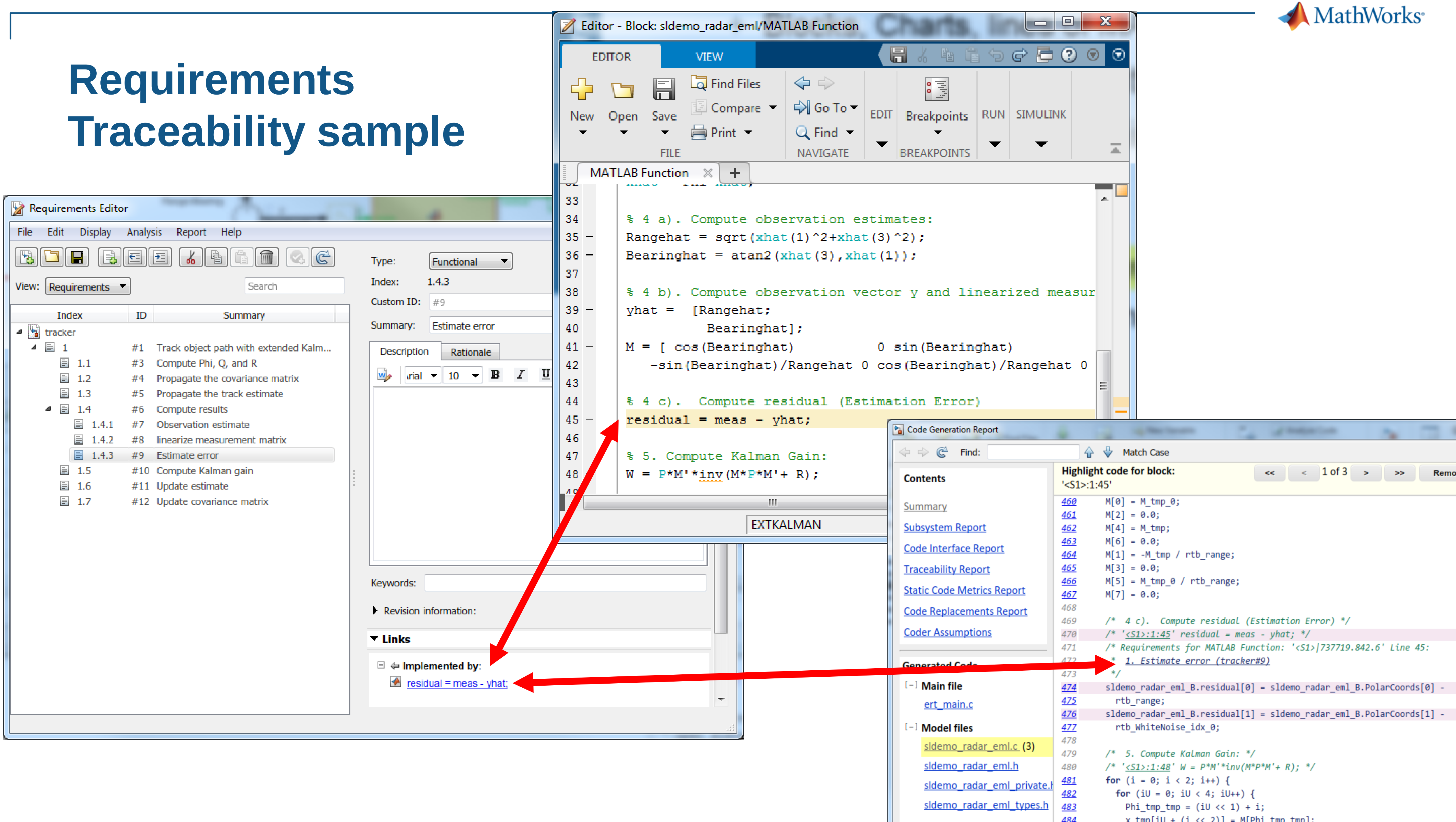
This Photo by Unknown Author is licensed under CC BY-SA

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



Index	ID	Summary
tracker		
1	#1	Track object path with extended Kalm...
1.1	#3	Compute Phi, Q, and R
1.2	#4	Propagate the covariance matrix
1.3	#5	Propagate the track estimate
1.4	#6	Compute results
1.4.1	#7	Observation estimate
1.4.2	#8	linearize measurement matrix
1.4.3	#9	Estimate error
1.5	#10	Compute Kalman gain
1.6	#11	Update estimate
1.7	#12	Update covariance matrix

Type: Functional
Index: 1.4.3
Custom ID: #9
Summary: Estimate error

Implemented by:
residual = meas - yhat;

```
% 4 a). Compute observation estimates:
Rangehat = sqrt(xhat(1)^2+xhat(3)^2);
Bearinghat = atan2(xhat(3),xhat(1));

% 4 b). Compute observation vector y and linearized measur
yhat =  [Rangehat;
         Bearinghat];
M = [ cos(Bearinghat)           0 sin(Bearinghat)
   -sin(Bearinghat)/Rangehat 0 cos(Bearinghat)/Rangehat 0

% 4 c).  Compute residual (Estimation Error)
residual = meas - yhat;

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

Highlight code for block:
'<S1>:1:45'

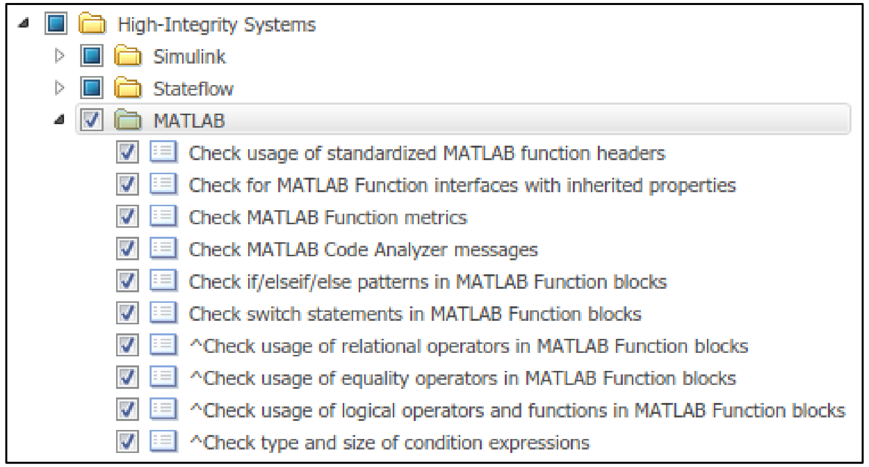
```
    M[0] = M_tmp_0;
    M[2] = 0.0;
    M[4] = M_tmp;
    M[6] = 0.0;
    M[1] = -M_tmp / rtb_range;
    M[3] = 0.0;
    M[5] = M_tmp_0 / rtb_range;
    M[7] = 0.0;

    /*  4 c).  Compute residual (Estimation Error) */
    /* '<S1>:1:45' residual = meas - yhat; */
    /* Requirements for MATLAB Function: '<S1>|737719.842.6' Line 45:
     *  1. Estimate error (tracker#9)
     */
    sldemo_radar_eml_B.residual[0] = sldemo_radar_eml_B.PolarCoords[0] -
      rtb_range;
    sldemo_radar_eml_B.residual[1] = sldemo_radar_eml_B.PolarCoords[1] -
      rtb_WhiteNoise_idx_0;

    /*  5. Compute Kalman Gain: */
    /* '<S1>:1:48' W = P*M'*inv(M*P*M'+ R); */
    for (i = 0; i < 2; i++) {
      for (iU = 0; iU < 4; iU++) {
        Phi_tmp_tmp = (iU << 1) + i;
```

Design and Code Standards

Simulink Check



- ■ High-Integrity Systems
  - ■ Simulink
  - ■ Stateflow
  - ☑ MATLAB
    - ☑ Check usage of standardized MATLAB function headers
    - ☑ Check for MATLAB Function interfaces with inherited properties
    - ☑ Check MATLAB Function metrics
    - ☑ Check MATLAB Code Analyzer messages
    - ☑ Check if/elseif/else patterns in MATLAB Function blocks
    - ☑ Check switch statements in MATLAB Function blocks
    - ☑ ^Check usage of relational operators in MATLAB Function blocks
    - ☑ ^Check usage of equality operators in MATLAB Function blocks
    - ☑ ^Check usage of logical operators and functions in MATLAB Function blocks
    - ☑ ^Check type and size of condition expressions

+ 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

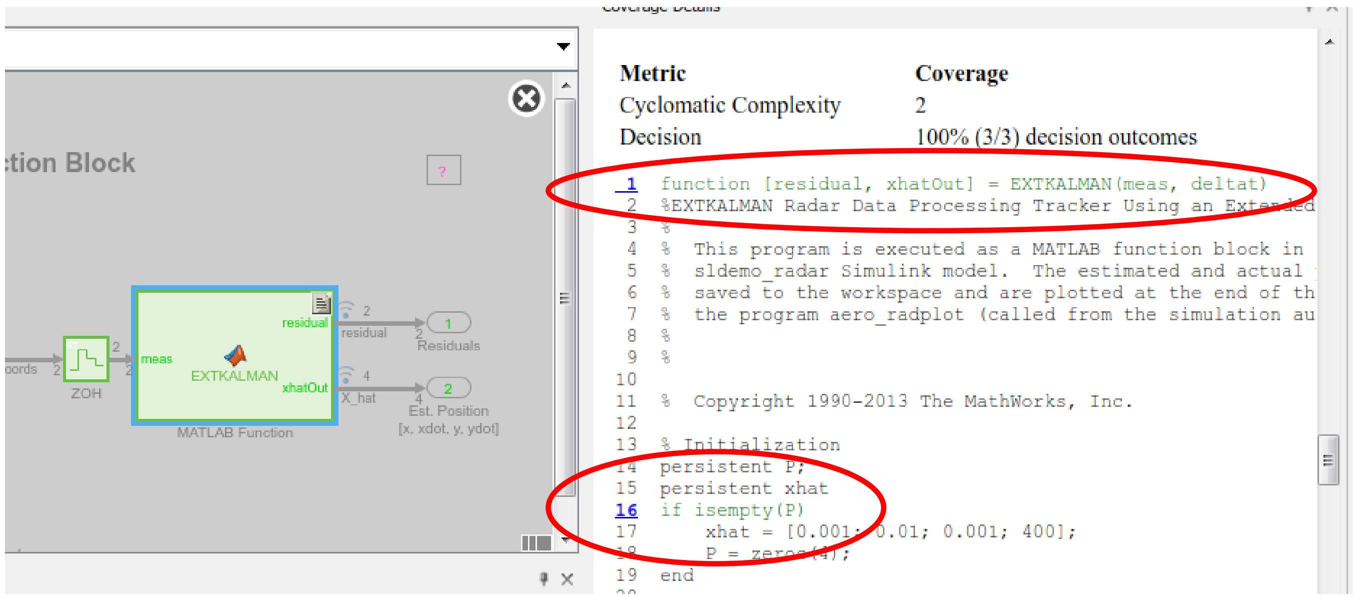
- Requirements-based test authoring, execution via Simulink Test
- Simulink Design Verifier (SLDV) property proving
- SLDV design error detection
- Back to back testing for model to code for Software-in-the-Loop (SIL), Processor-in-the-Loop (PIL)

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



Metric	Coverage
Cyclomatic Complexity	2
Decision	100% (3/3) decision outcomes

```
function [residual, xhatOut] = EXTKALMAN(meas, deltat)
%EXTKALMAN Radar Data Processing Tracker Using an Extended
%
%  This program is executed as a MATLAB function block in
%  sldemo_radar Simulink model.  The estimated and actual
%  saved to the workspace and are plotted at the end of th
%  the program aero_radplot (called from the simulation au
%
%

%  Copyright 1990-2013 The MathWorks, Inc.

% Initialization
persistent P;
persistent xhat
if isempty(P)
    xhat = [0.001; 0.01; 0.001; 400];
    P = zeros(4);
end
```

Summary so far

- Customers are successfully using MATLAB in ISO 26262-compliant products today

- Our verification workflow and tools support MATLAB called by Simulink

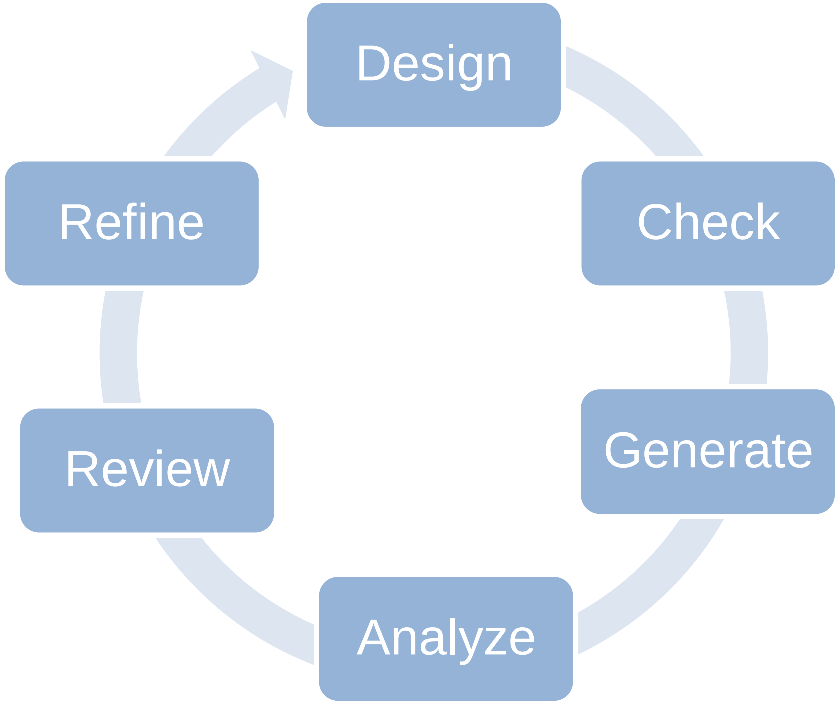
- But… there are some gaps remaining
  - Potential issues with MISRA-C compliance of code generated from MATLAB
  - Achieving MATLAB vs C code coverage
  - Simplifying Simulink model ~~comparison~~ reviews

Code standards compliance

- Practice is to
  - run model checks
  - generate code
  - analyze compliance

Simulink Check

Polyspace Bug Finder

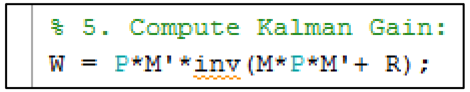


- Issues discovered?
  - document and proceed
  - rework the algorithm
  - rewrite a compliant function (toolboxes)

- Result is an allowed function list (*language subset*)
- Process gets more efficient over time

Code coverage

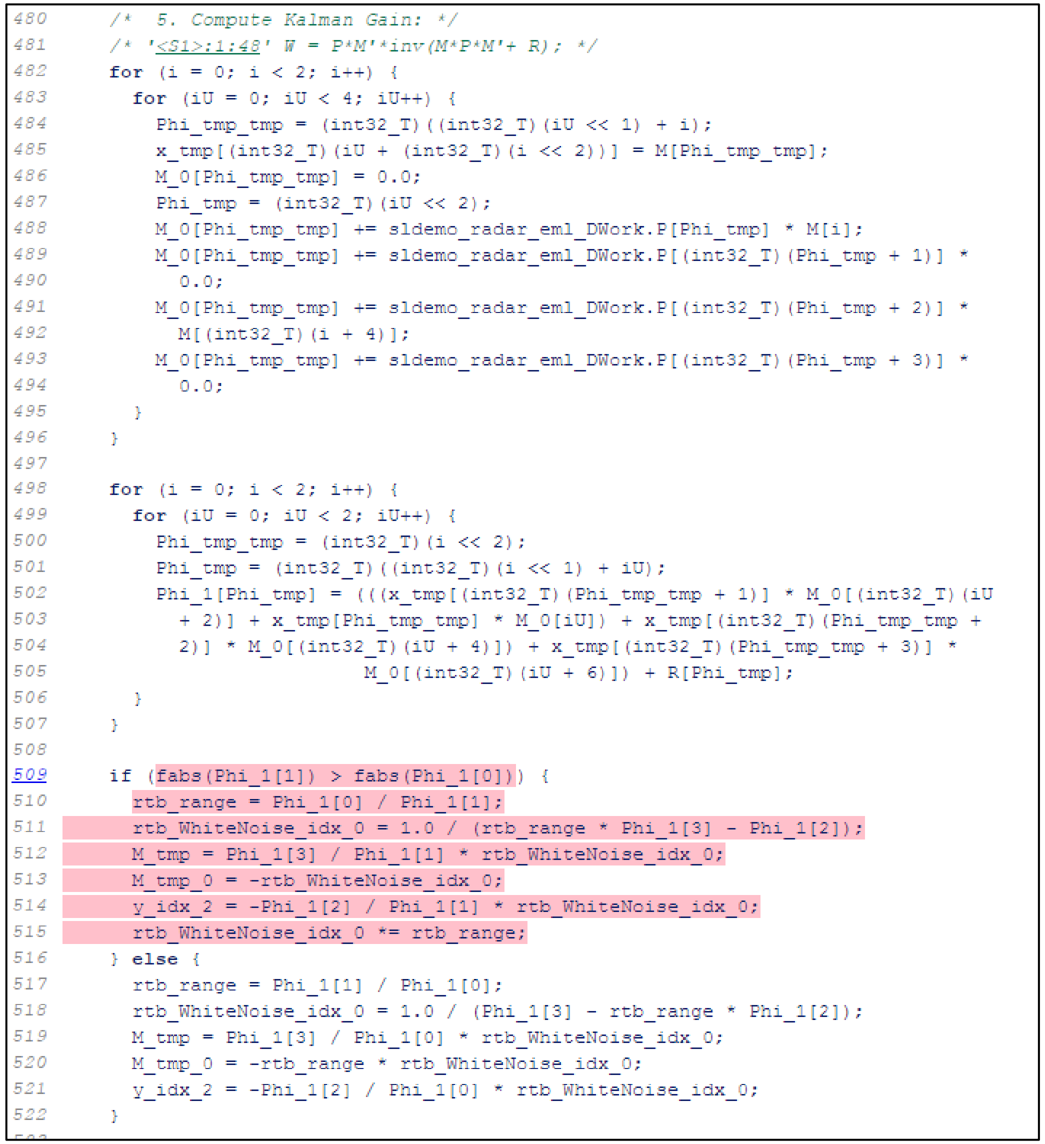
- MATLAB functions can be complex in C/C++



```
% 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



```
  /*  5. Compute Kalman Gain: */
  /* '<S1>:1:48' W = P*M'*inv(M*P*M'+ R); */
  for (i = 0; i < 2; i++) {
    for (iU = 0; iU < 4; iU++) {
      Phi_tmp_tmp = (int32_T)((int32_T)(iU << 1) + i);
      x_tmp[(int32_T)(iU + (int32_T)(i << 2))] = M[Phi_tmp_tmp];
      M_0[Phi_tmp_tmp] = 0.0;
      Phi_tmp = (int32_T)(iU << 2);
      M_0[Phi_tmp_tmp] += sldemo_radar_eml_DWork.P[Phi_tmp] * M[i];
      M_0[Phi_tmp_tmp] += sldemo_radar_eml_DWork.P[(int32_T)(Phi_tmp + 1)] *
        0.0;
      M_0[Phi_tmp_tmp] += sldemo_radar_eml_DWork.P[(int32_T)(Phi_tmp + 2)] *
        M[(int32_T)(i + 4)];
      M_0[Phi_tmp_tmp] += sldemo_radar_eml_DWork.P[(int32_T)(Phi_tmp + 3)] *
        0.0;
    }
  }

  for (i = 0; i < 2; i++) {
    for (iU = 0; iU < 2; iU++) {
      Phi_tmp_tmp = (int32_T)(i << 2);
      Phi_tmp = (int32_T)((int32_T)(i << 1) + iU);
      Phi_1[Phi_tmp] = (((x_tmp[(int32_T)(Phi_tmp_tmp + 1)] * M_0[(int32_T)(iU
        + 2)] + x_tmp[Phi_tmp_tmp] * M_0[iU]) + x_tmp[(int32_T)(Phi_tmp_tmp +
        2)] * M_0[(int32_T)(iU + 4)]) + x_tmp[(int32_T)(Phi_tmp_tmp + 3)] *
                        M_0[(int32_T)(iU + 6)]) + R[Phi_tmp];
    }
  }

  if (fabs(Phi_1[1]) > fabs(Phi_1[0])) {
    rtb_range = Phi_1[0] / Phi_1[1];
    rtb_WhiteNoise_idx_0 = 1.0 / (rtb_range * Phi_1[3] - Phi_1[2]);
    M_tmp = Phi_1[3] / Phi_1[1] * rtb_WhiteNoise_idx_0;
    M_tmp_0 = -rtb_WhiteNoise_idx_0;
    y_idx_2 = -Phi_1[2] / Phi_1[1] * rtb_WhiteNoise_idx_0;
    rtb_WhiteNoise_idx_0 *= rtb_range;
  } else {
    rtb_range = Phi_1[1] / Phi_1[0];
    rtb_WhiteNoise_idx_0 = 1.0 / (Phi_1[3] - rtb_range * Phi_1[2]);
    M_tmp = Phi_1[3] / Phi_1[0] * rtb_WhiteNoise_idx_0;
    M_tmp_0 = -rtb_range * rtb_WhiteNoise_idx_0;
    y_idx_2 = -Phi_1[2] / Phi_1[0] * rtb_WhiteNoise_idx_0;
  }
```

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

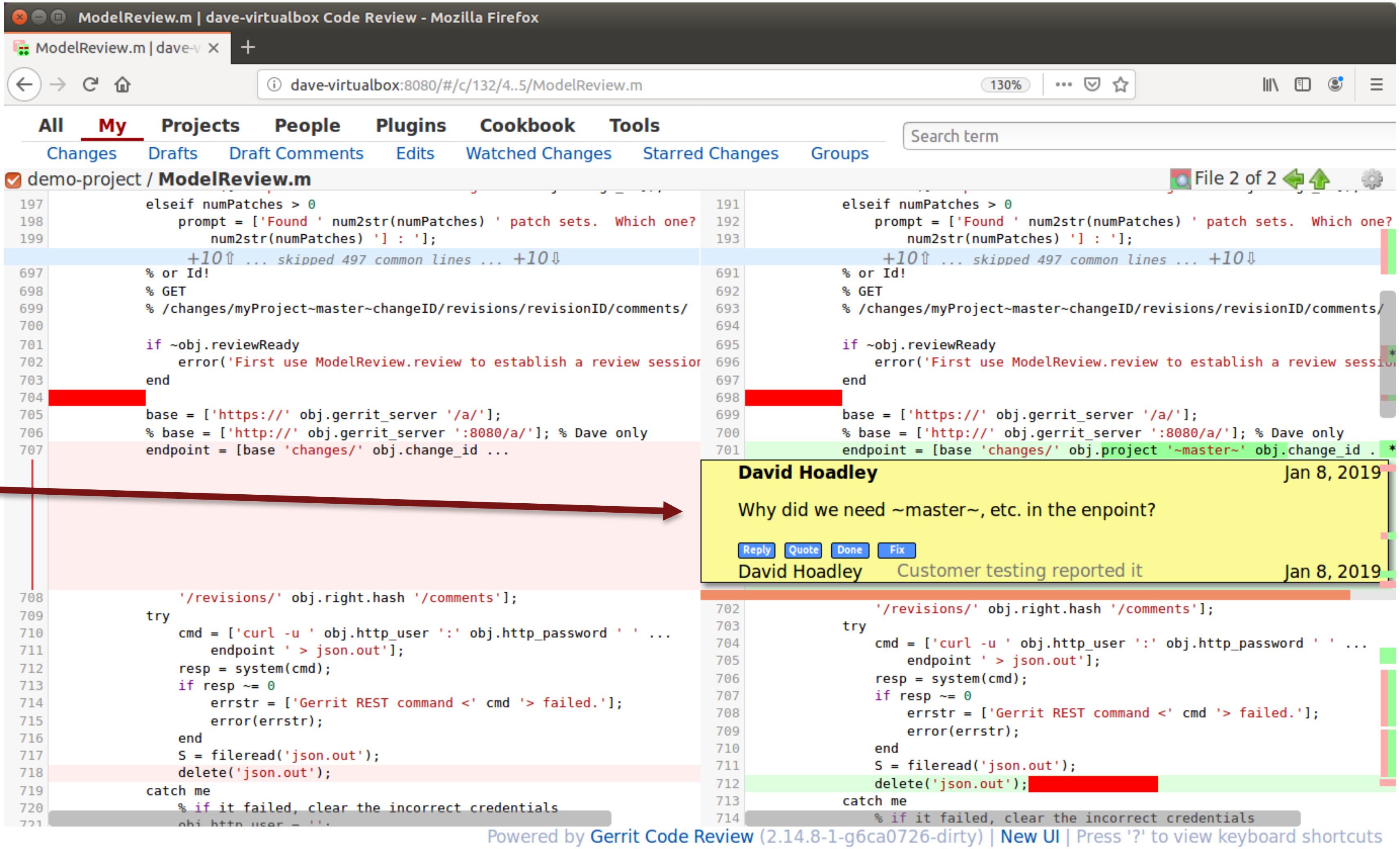
Text-based differences + review comments

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)



```
        elseif numPatches > 0
            prompt = ['Found ' num2str(numPatches) ' patch sets.  Which one?
                num2str(numPatches) '] : '];
          +10⇧ ... skipped 497 common lines ... +10⇩
        % or Id!
        % GET
        % /changes/myProject~master~changeID/revisions/revisionID/comments/

        if ~obj.reviewReady
            error('First use ModelReview.review to establish a review sessior
        end

        base = ['https://' obj.gerrit_server '/a/'];
        % base = ['http://' obj.gerrit_server ':8080/a/']; % Dave only
        endpoint = [base 'changes/' obj.change_id ...
            '/revisions/' obj.right.hash '/comments'];
        try
            cmd = ['curl -u ' obj.http_user ':' obj.http_password ' ' ...
                endpoint ' > json.out'];
            resp = system(cmd);
            if resp ~= 0
                errstr = ['Gerrit REST command <' cmd '> failed.'];
                error(errstr);
            end
            S = fileread('json.out');
            delete('json.out');
        catch me
            % if it failed, clear the incorrect credentials
            obj.http_user = '';
```

```
        elseif numPatches > 0
            prompt = ['Found ' num2str(numPatches) ' patch sets.  Which one?
                num2str(numPatches) '] : '];
          +10⇧ ... skipped 497 common lines ... +10⇩
        % or Id!
        % GET
        % /changes/myProject~master~changeID/revisions/revisionID/comments/

        if ~obj.reviewReady
            error('First use ModelReview.review to establish a review sessio
        end

        base = ['https://' obj.gerrit_server '/a/'];
        % base = ['http://' obj.gerrit_server ':8080/a/']; % Dave only
        endpoint = [base 'changes/' obj.project '~master~' obj.change_id .
```

David Hoadley Jan 8, 2019

Why did we need ~master~, etc. in the enpoint?

Reply Quote Done Fix

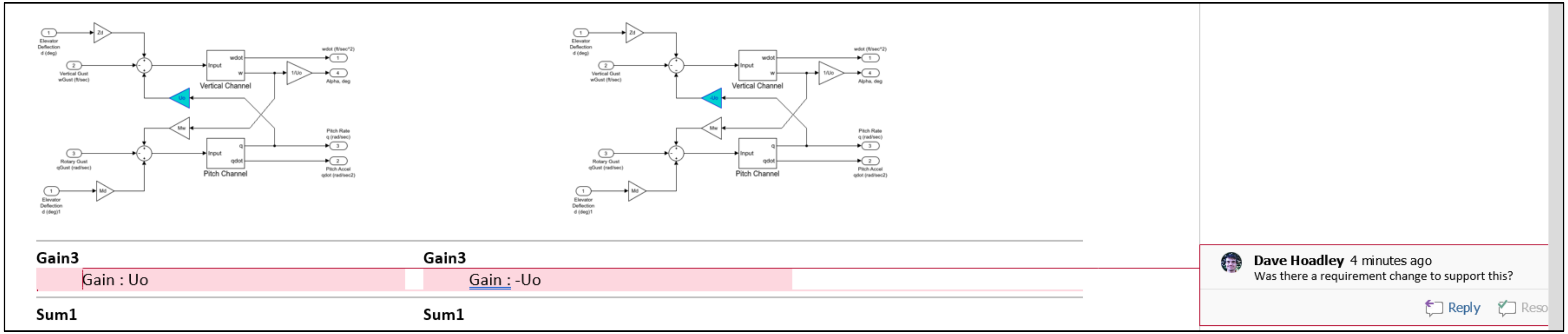
David Hoadley Customer testing reported it Jan 8, 2019

```
            '/revisions/' obj.right.hash '/comments'];
        try
            cmd = ['curl -u ' obj.http_user ':' obj.http_password ' ' ...
                endpoint ' > json.out'];
            resp = system(cmd);
            if resp ~= 0
                errstr = ['Gerrit REST command <' cmd '> failed.'];
                error(errstr);
            end
            S = fileread('json.out');
            delete('json.out');
        catch me
            % if it failed, clear the incorrect credentials
```

Powered by Gerrit Code Review (2.14.8-1-g6ca0726-dirty) | New UI | Press '?' to view keyboard shortcuts

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

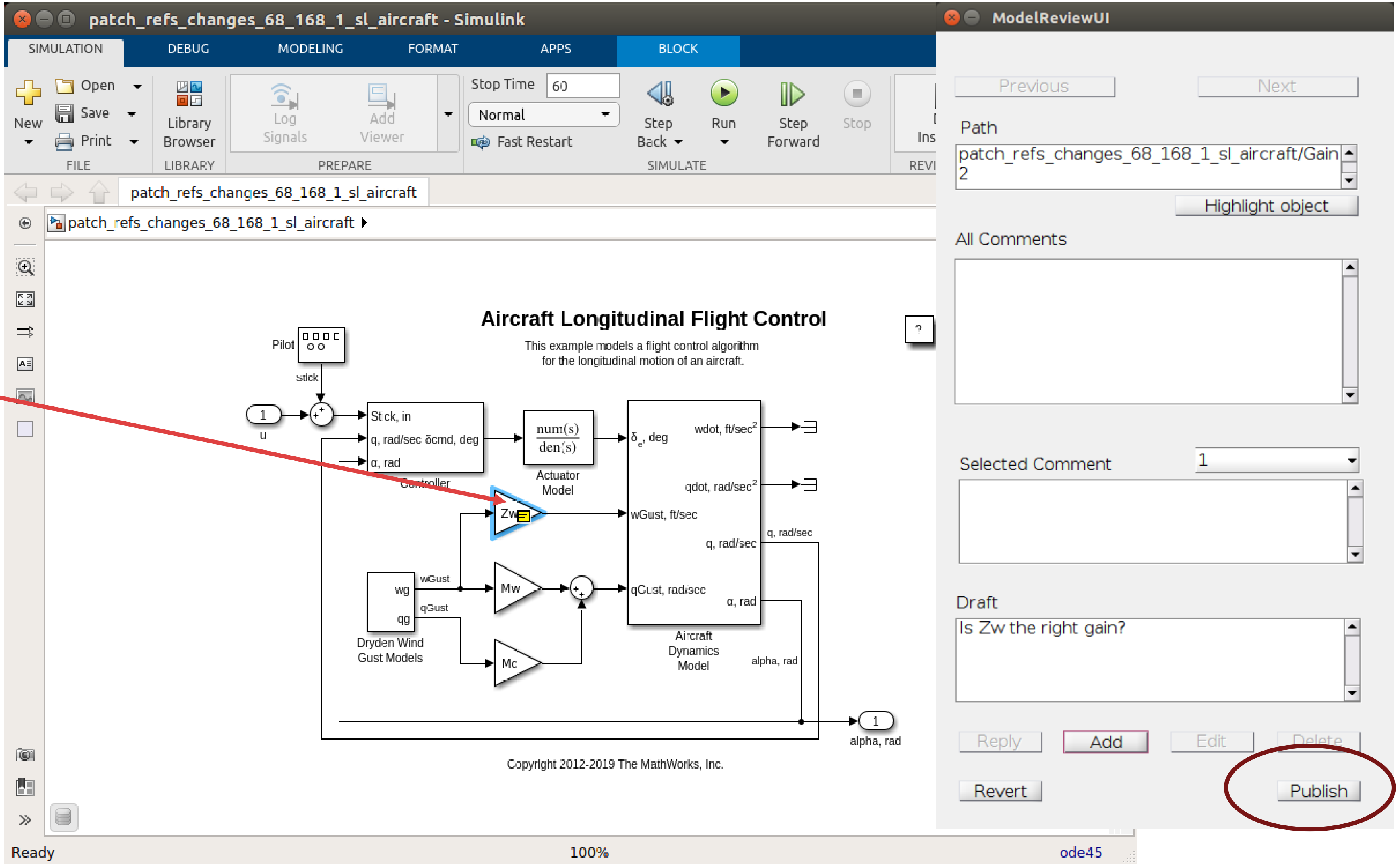

1
Elevator Deflection d (deg)
Zd
2
Vertical Gust wGust (ft/sec)
Input
wdot
w
Vertical Channel
1/Uo
wdot (ft/sec^2)
1
4
Alpha, deg
Uo
-Uo
Mw
3
Rotary Gust qGust (rad/sec)
Input
q
qdot
Pitch Channel
Pitch Rate q (rad/sec)
3
2
Pitch Accel qdot (rad/sec2)
1
Elevator Deflection d (deg)1
Md
Reply
Reso

Gain3	Gain3
Gain : Uo	Gain : -Uo
Sum1	Sum1

Dave Hoadley 4 minutes ago
Was there a requirement change to support this?

Extending this concept *into* Simulink

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



patch_refs_changes_68_168_1_sl_aircraft - Simulink

Stop Time: 60

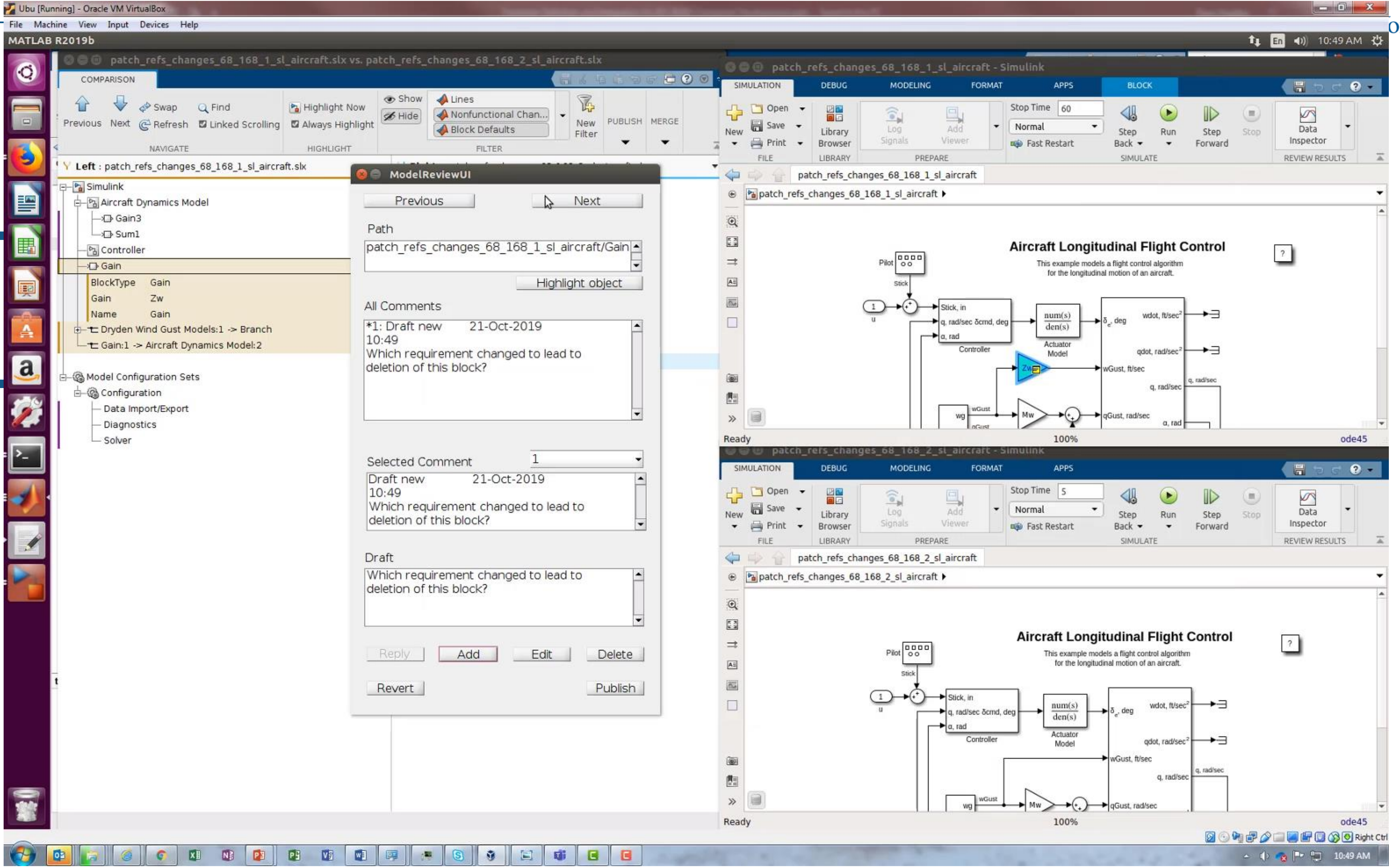
Normal

ModelReviewUI

Path: patch_refs_changes_68_168_1_sl_aircraft/Gain 2

Selected Comment: 1

Draft: Is Zw the right gain?



Ubu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MATLAB R2019b
10:49 AM
COMPARISON
Previous Next Swap Refresh Find Linked Scrolling
Highlight Now Always Highlight
Show Hide
Lines Nonfunctional Chan... Block Defaults
New Filter
PUBLISH MERGE
NAVIGATE HIGHLIGHT FILTER
SIMULATION DEBUG MODELING FORMAT APPS BLOCK
Open Save Print New
Library Browser
Log Signals Add Viewer
Step Back Run Step Forward Stop
Fast Restart
Data Inspector
FILE LIBRARY PREPARE SIMULATE REVIEW RESULTS
Aircraft Longitudinal Flight Control
This example models a flight control algorithm for the longitudinal motion of an aircraft.
Pilot
Stick
1
u
Stick, in
q, rad/sec δcmd, deg
α, rad
Controller
num(s)
den(s)
Actuator Model
δe, deg
wdot, ft/sec²
qdot, rad/sec²
wGust, ft/sec
q, rad/sec
qGust, rad/sec
α, rad
wg
wGust
Mw
Zw
Ready
100%
ode45
Right Ctrl

patch_refs_changes_68_168_1_sl_aircraft.slx vs. patch_refs_changes_68_168_2_sl_aircraft.slx

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
Stop Time 60 · Normal

patch_refs_changes_68_168_2_sl_aircraft - Simulink
Stop Time 5 · Normal

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