MATLAB EXPO

Collaborative Software Development in MATLAB and Simulink

Adam Sifounakis
MATLAB Language Product Manager

MathWorks®

How complex are your projects?

- Hundreds of files?
- Many file dependencies?
- Complex setup required?
- …?

How many people are involved in your project?

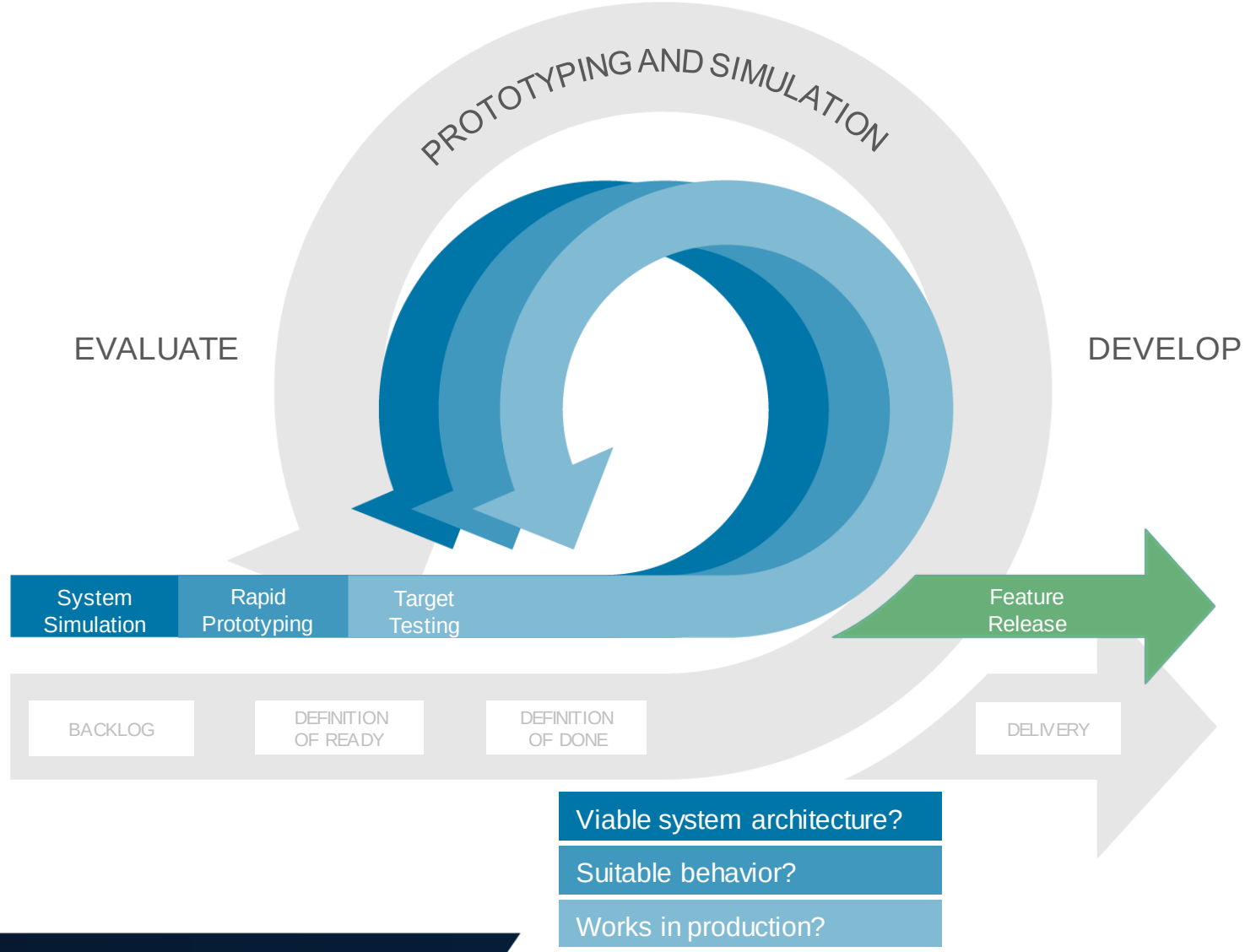
- Dozens of developers?
- Cross-disciplinary teams?
- Teams across the world?
- …?

How do you ensure project quality?

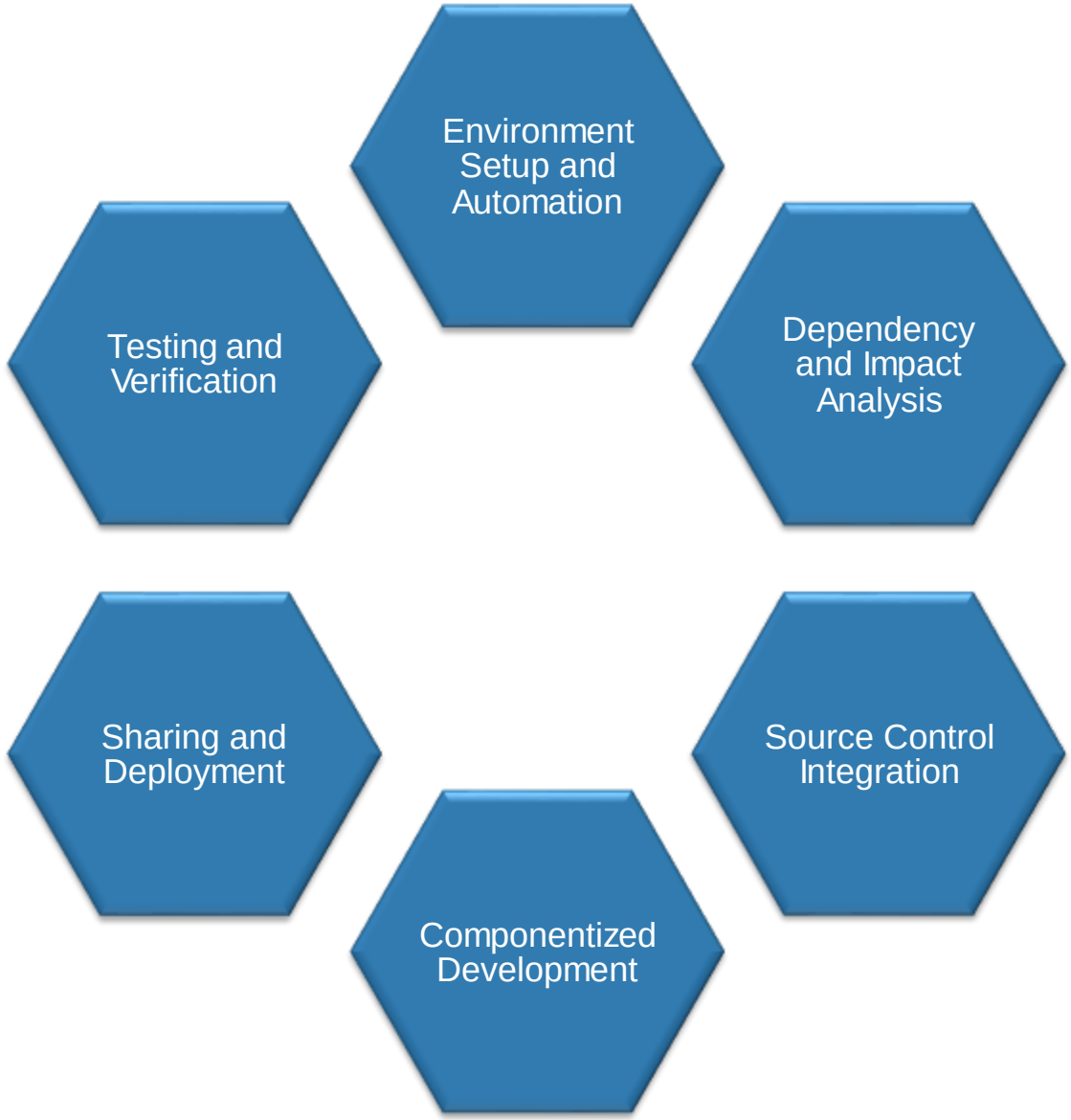
- Systematic testing?
- Coding/modeling standards?
- Regulatory oversight?
- Trust that it just works?
- …?

Develop quality software with MATLAB and Simulink

- Good software development practices help improve code and model quality
- The tools and practices we discuss today support Agile development workflows



Robust, collaborative development requires…



Robust, collaborative development requires…

Agenda

	Setting up your development environment
	Managing team workflows
	Developing better code and models
	Testing and verification

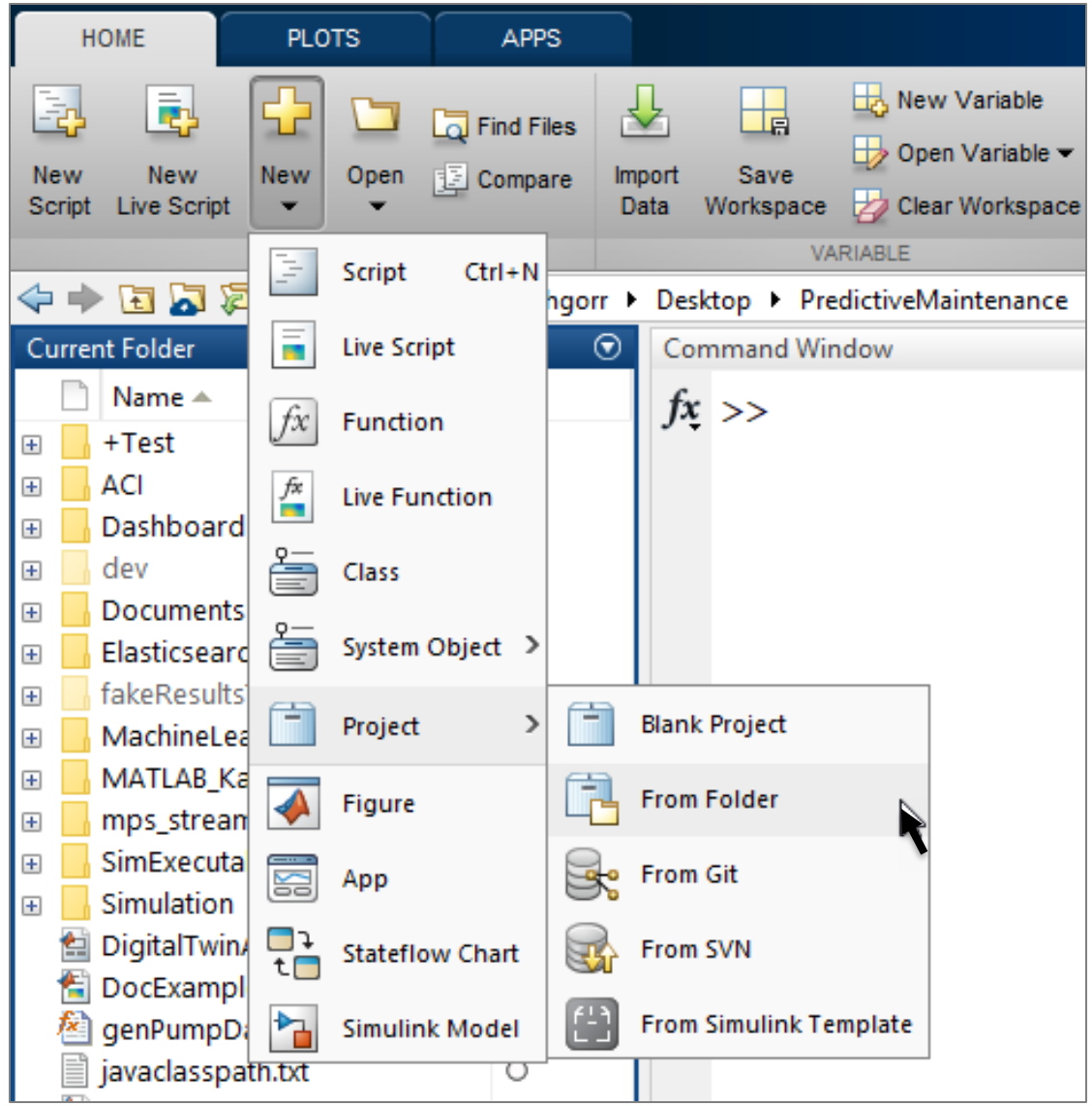
Development Challenges

"It works on my computer, but not on yours…"

- Incomplete set of files?
- Which files are missing?
- Different environment?
- How to get started with a project?
- …

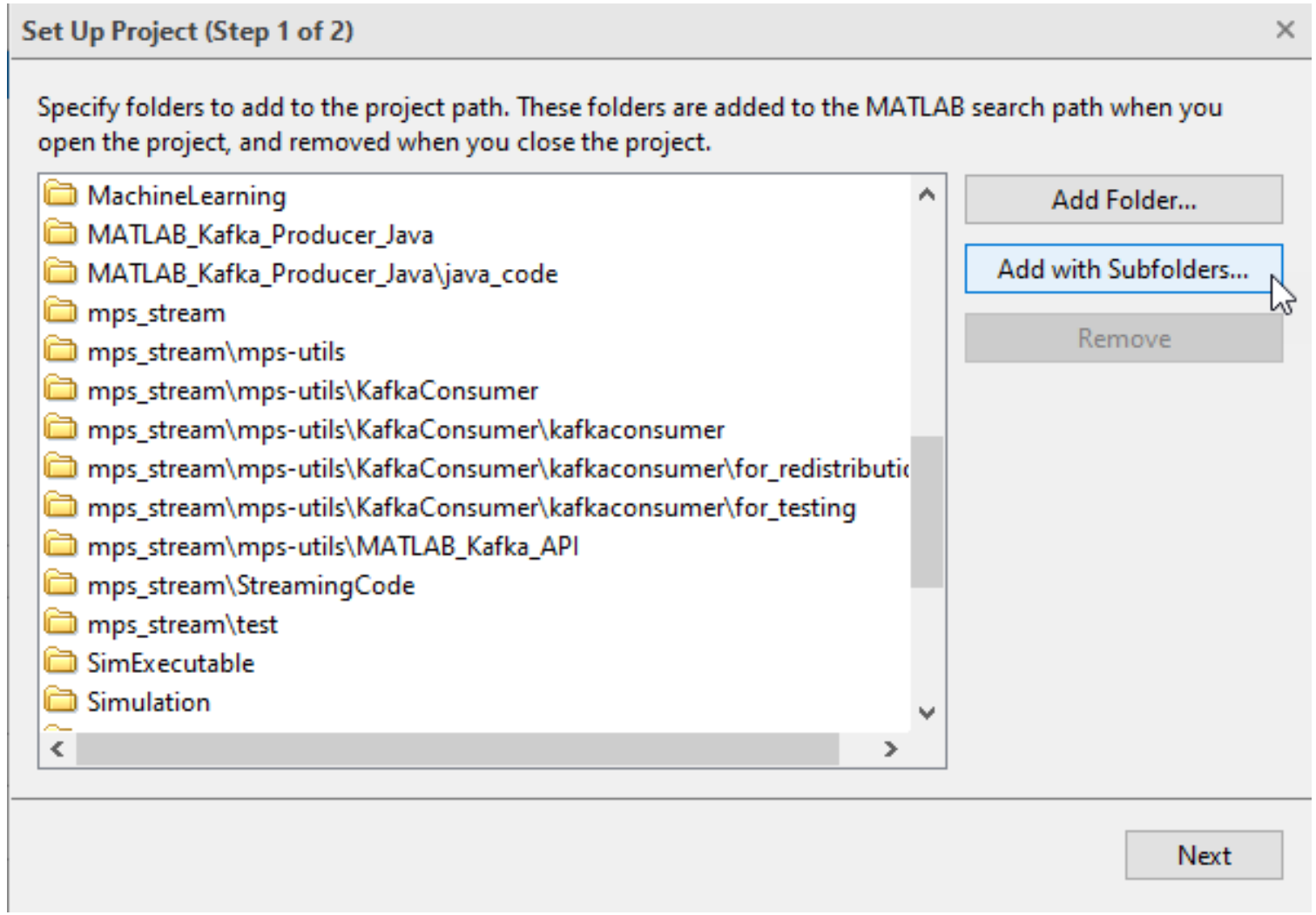
Managing Your Work with Projects

1. Create project



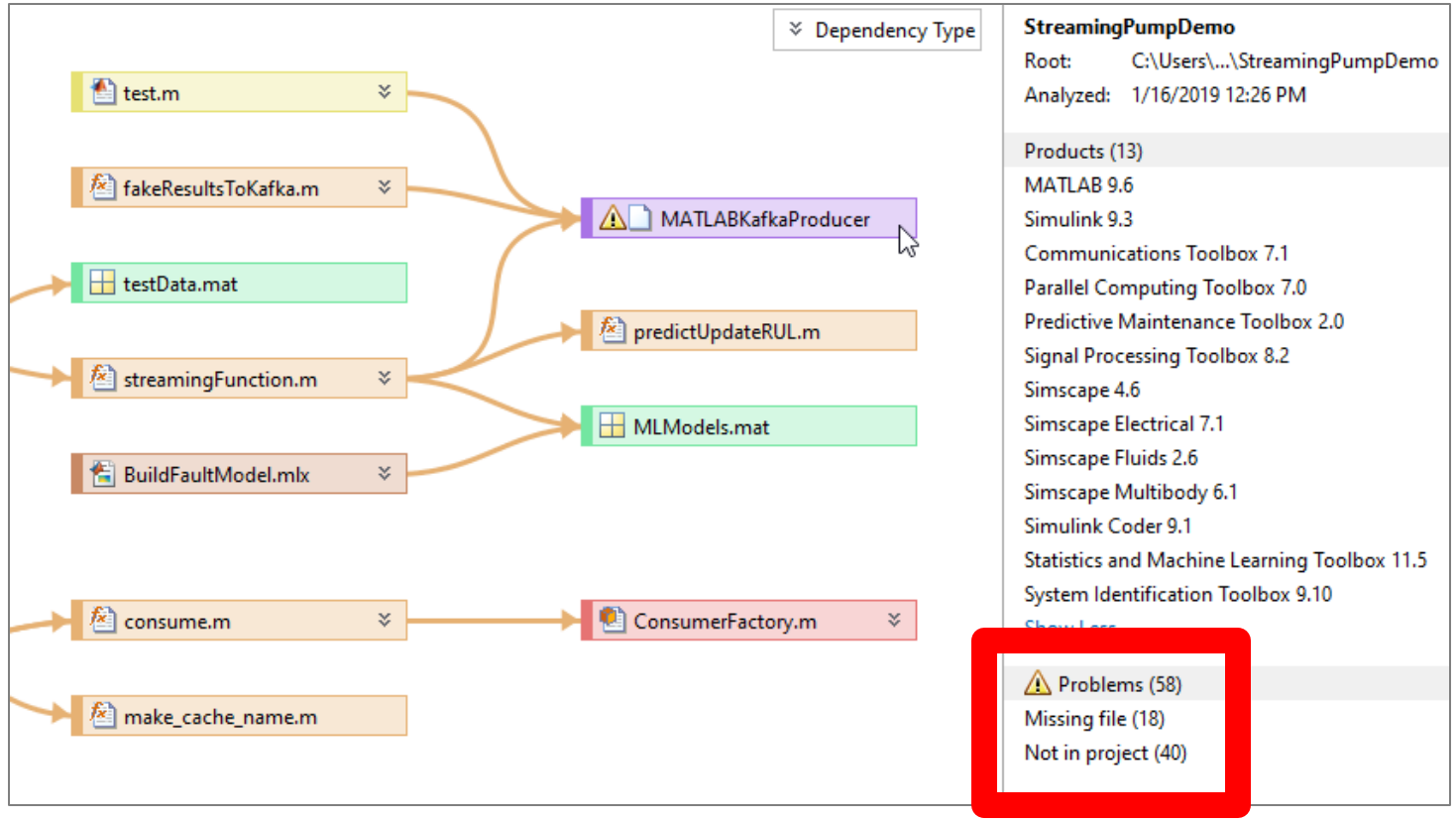
Managing Your Work with Projects

1. Create project
2. Set path and startup tasks



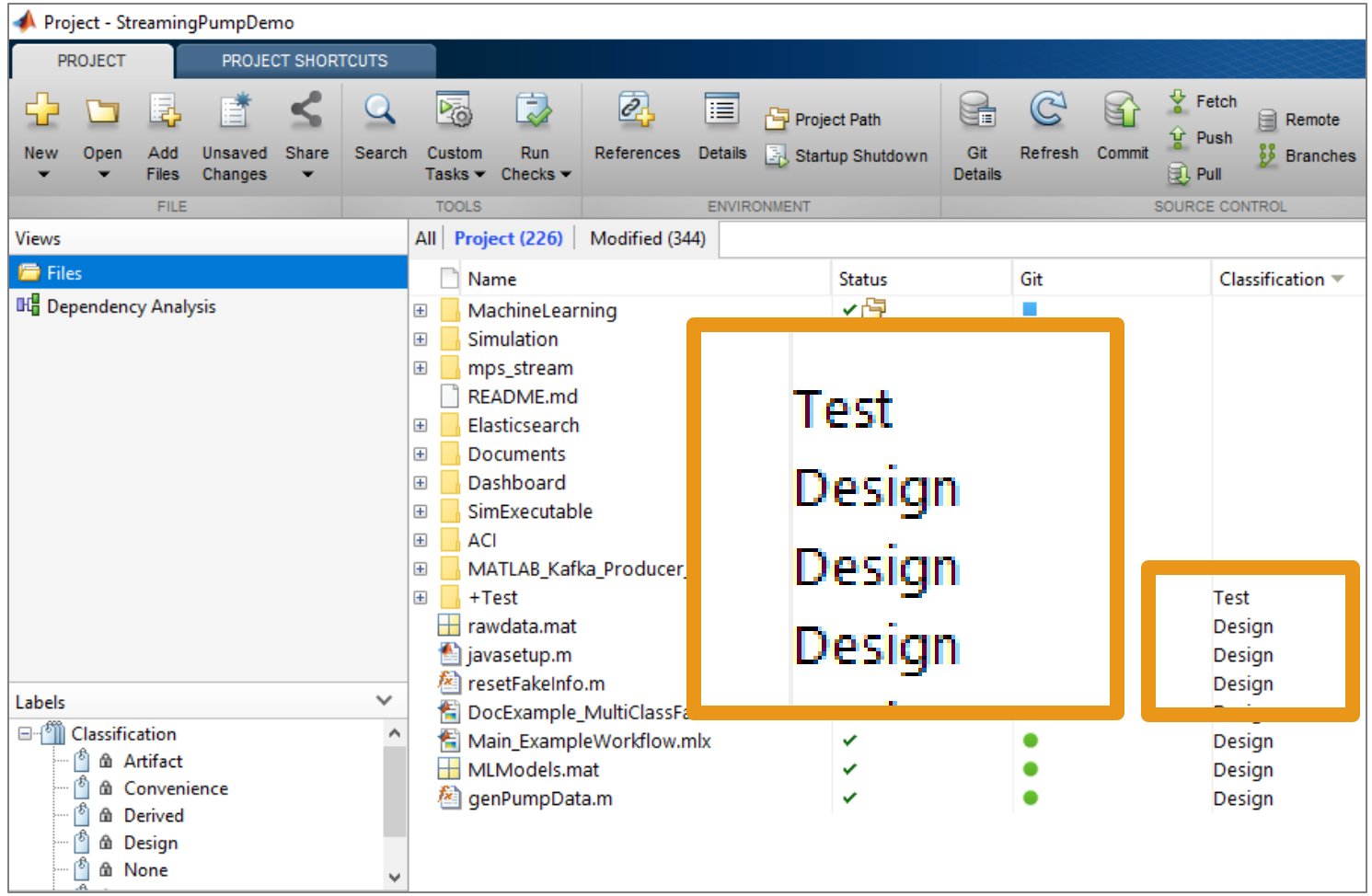
Managing Your Work with Projects

1. Create project
2. Set path and startup tasks
3. Explore dependencies



Managing Your Work with Projects

1. Create project
2. Set path and startup tasks
3. Explore dependencies
4. Label files

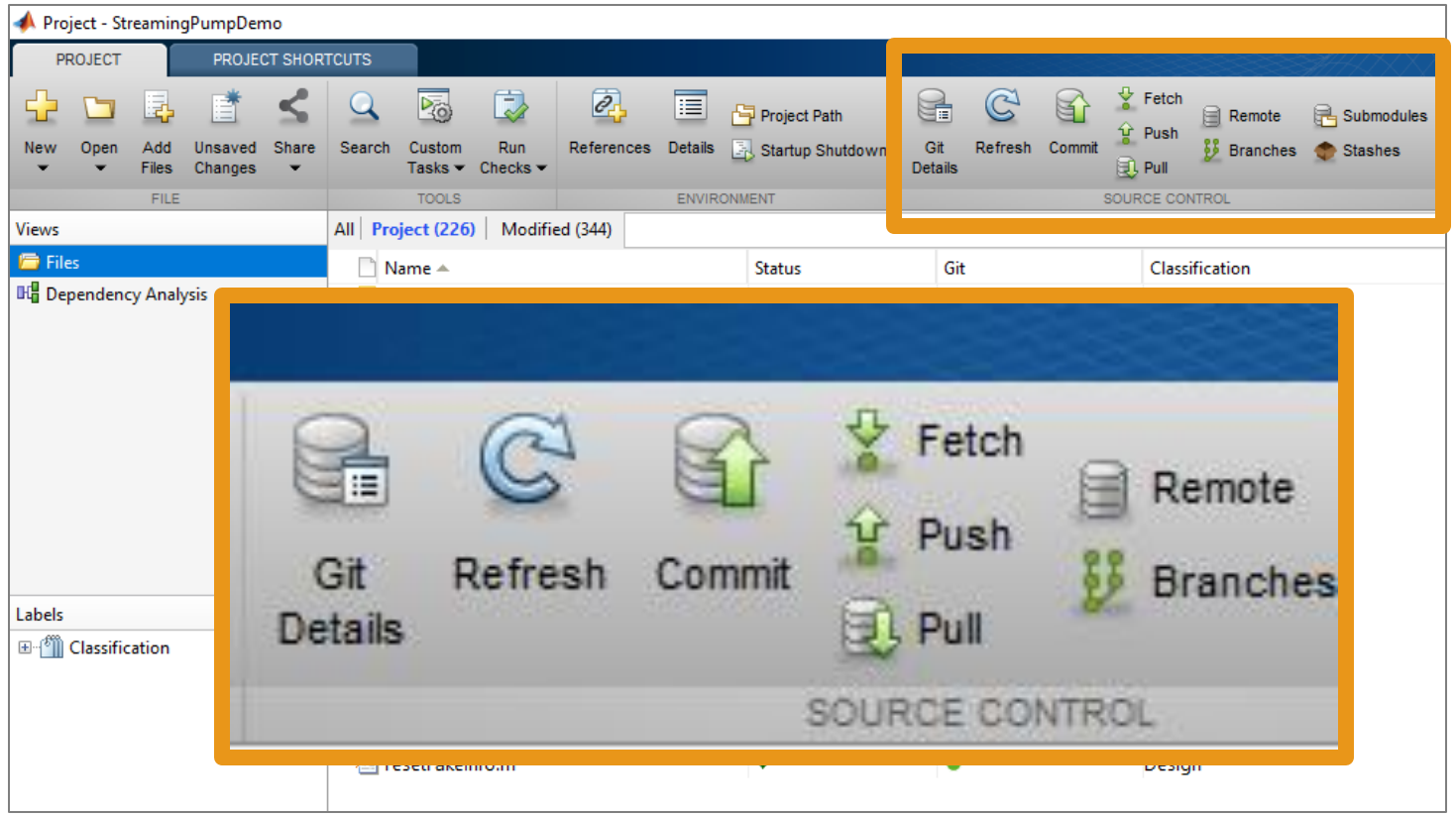


Identify and run tests locally

…and on Continuous Integration (CI) servers

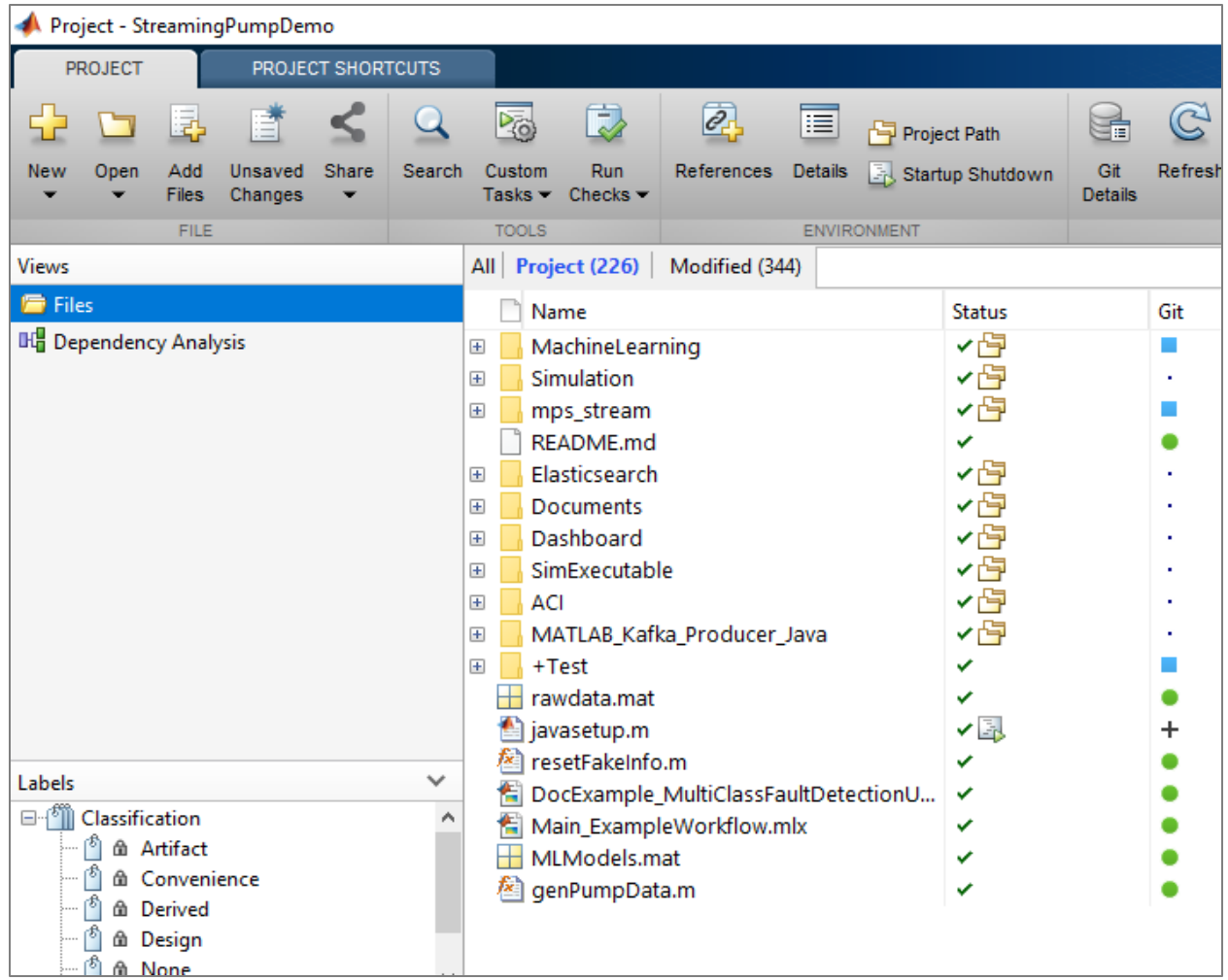
Managing Your Work with Projects

1. Create project
2. Set path and startup tasks
3. Explore dependencies
4. Label files
5. Integrate source control



Projects in MATLAB and Simulink

- Manage your files and path
- Analyze file dependencies
- Function refactoring
- Run startup & shutdown tasks
- Create project shortcuts
- Label and filter files
- Integrate source control



Agenda

	Setting up your development environment
	Managing team workflows
	Developing better code and models
	Testing and verification

Team-Based Development Challenges

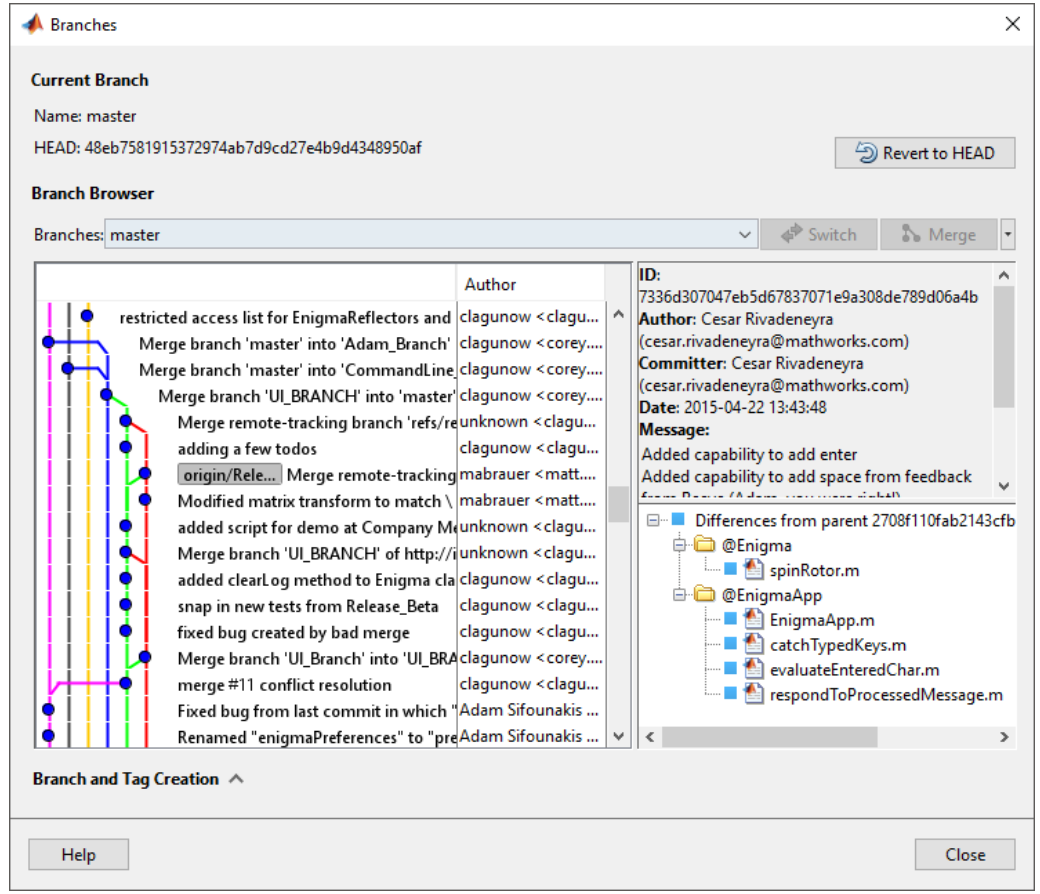
"Someone else broke my code…"

- Develop code without affecting others?
- Identify the source of development conflicts?
- Resolve development conflicts?
- …

Source Control

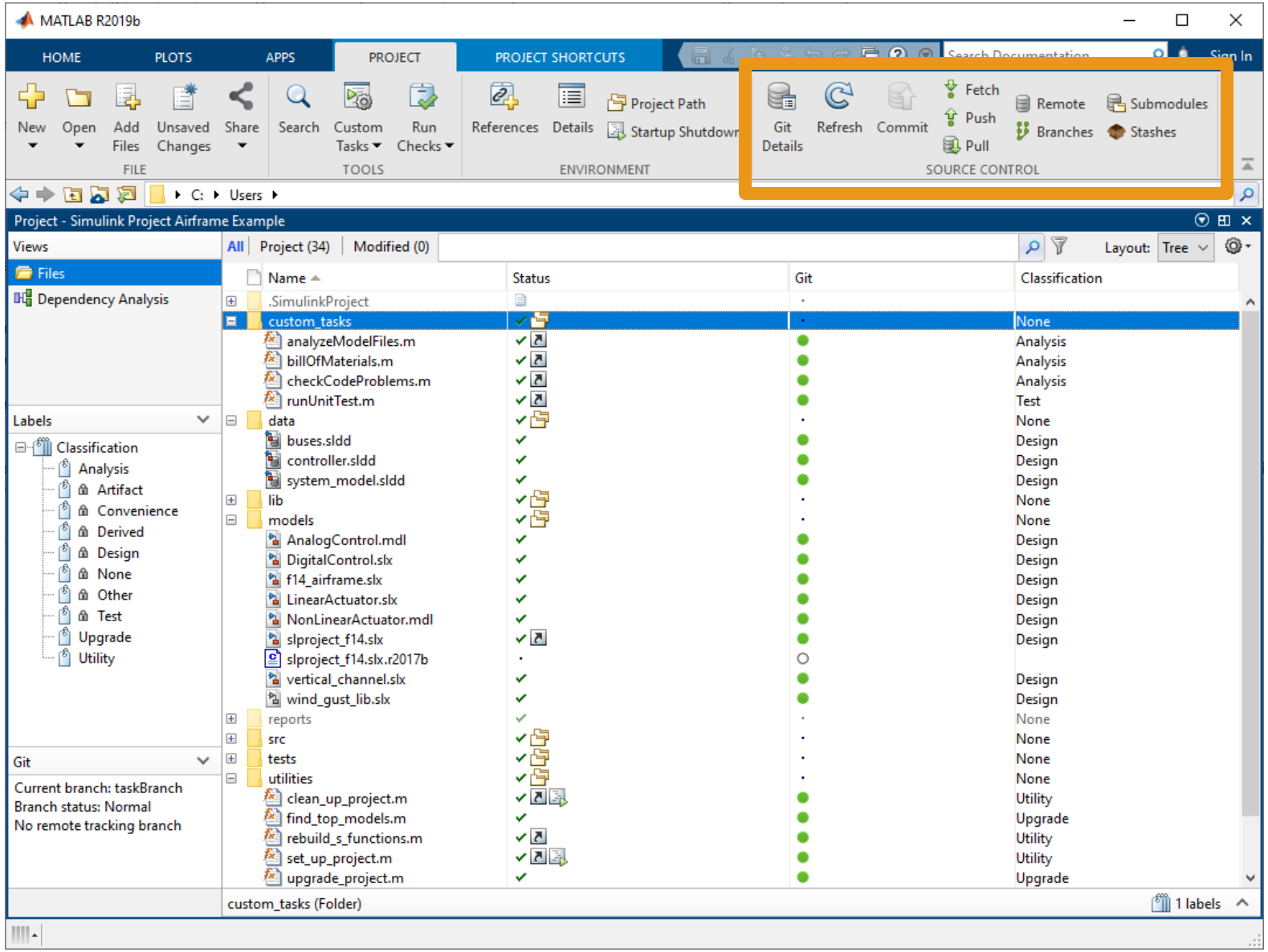
- A system to manage changes to code, models, documents, etc.

- Benefits of source control:
  - Maintain backups, history, and ability to restore
  - Track changes and responsibility
  - Reconcile conflicting changes
  - Generate discussion
  - Save you from yourself



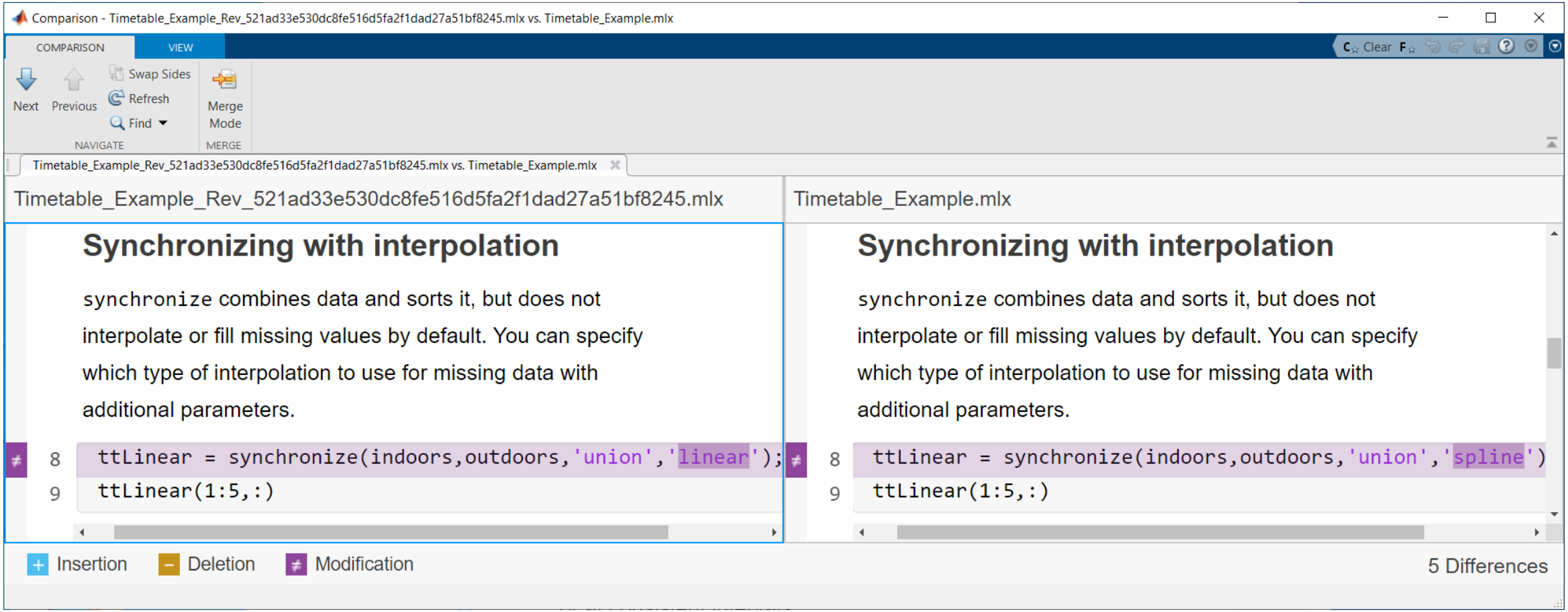
Source Control Integration

- Manage your code and models from within MATLAB and Simulink
- Git integrated into:
  - Projects
  - Current Folder browser
- Use Comparison Tool to view and merge changes between revisions



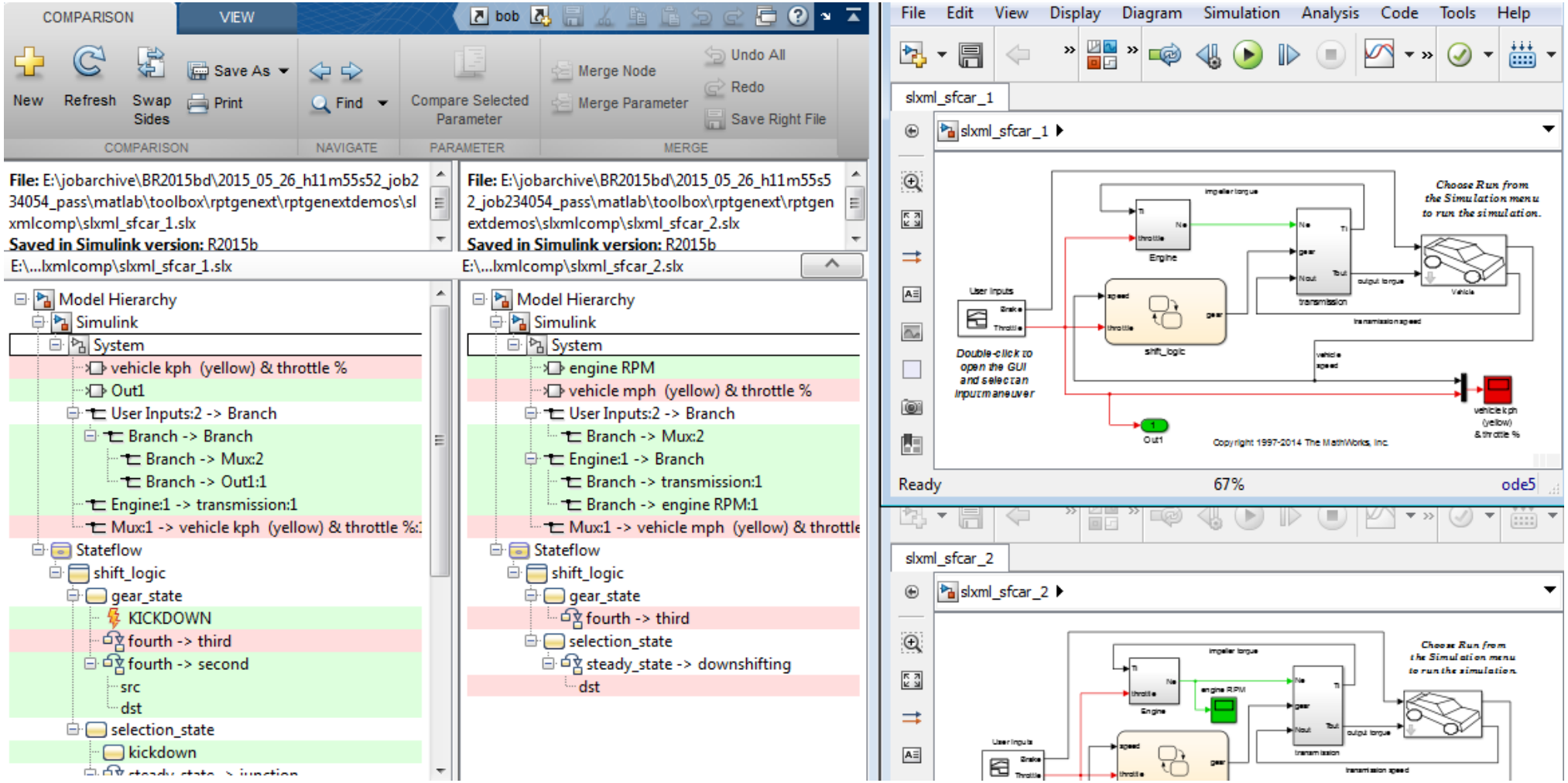
Comparison Tool and 3-way merge resolution

MATLAB



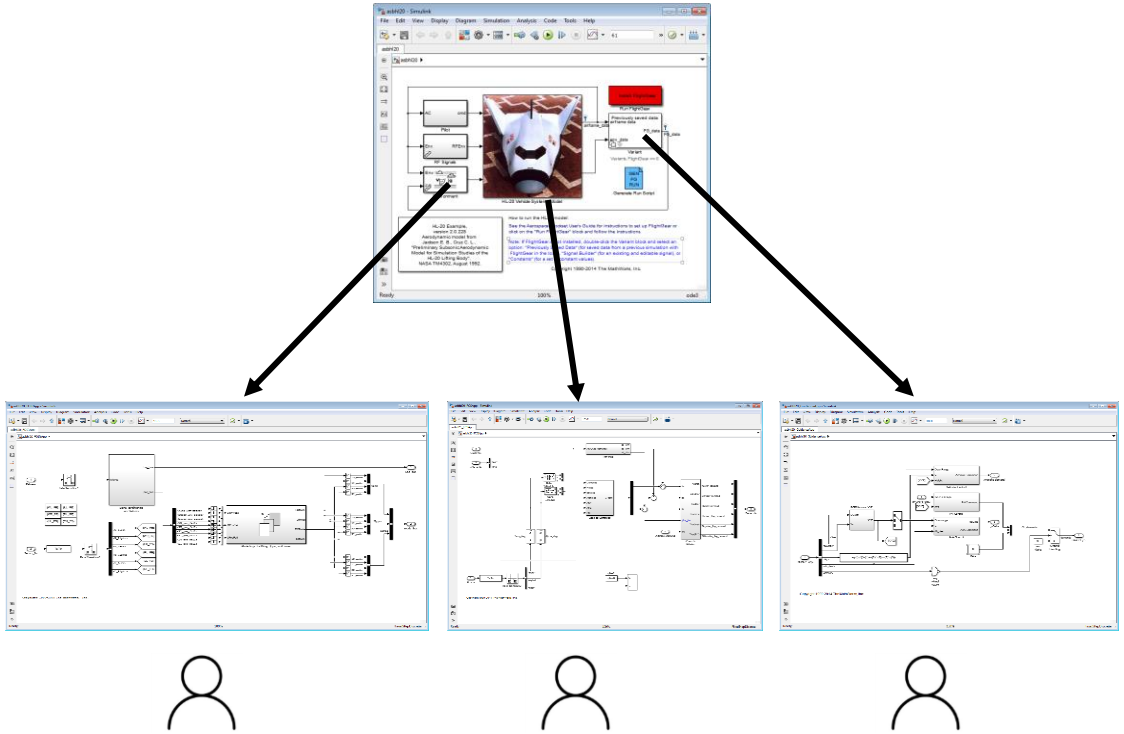
Comparison Tool and 3-way merge resolution

Simulink

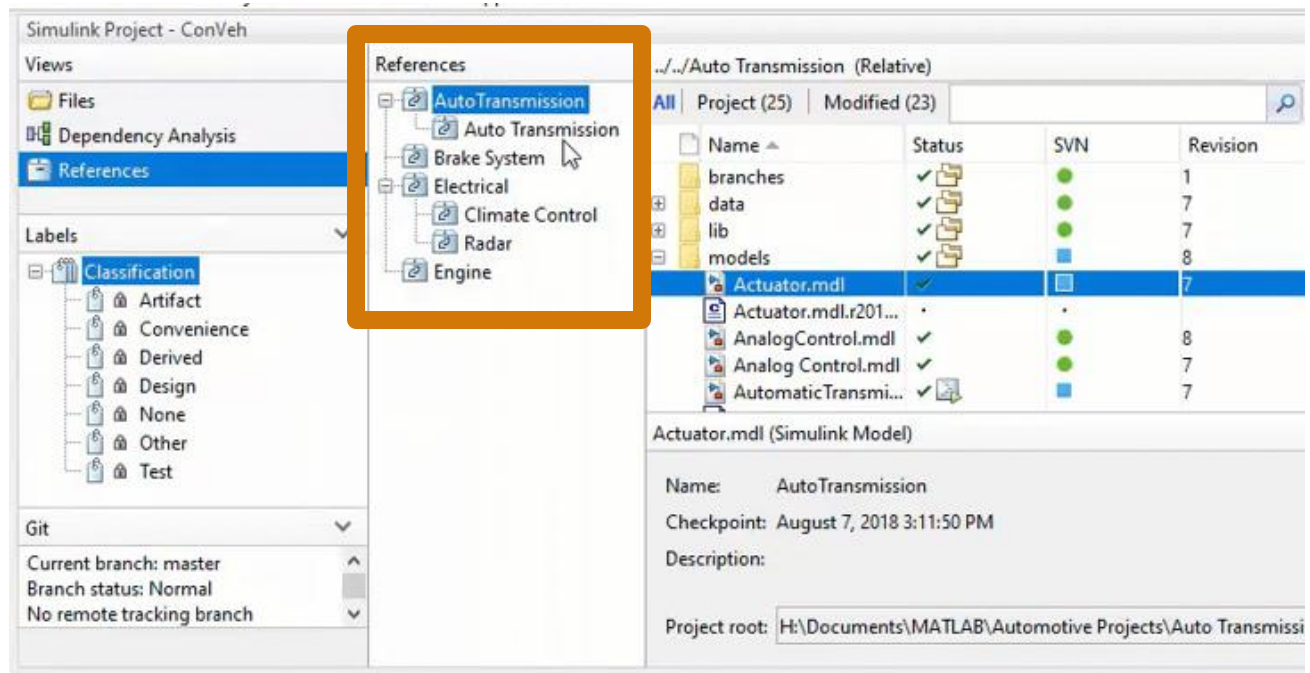


Managing Complexity with Model and Project References

Model Reference



Project Reference



Agenda

	Setting up your development environment
	Managing team workflows
	Developing better code and models
	Testing and verification

What defines a “better” design?

- Faster?
- More memory efficient?
- Better organized?
- More stable?
- More portable?
- Easier to maintain?
- …

YES!

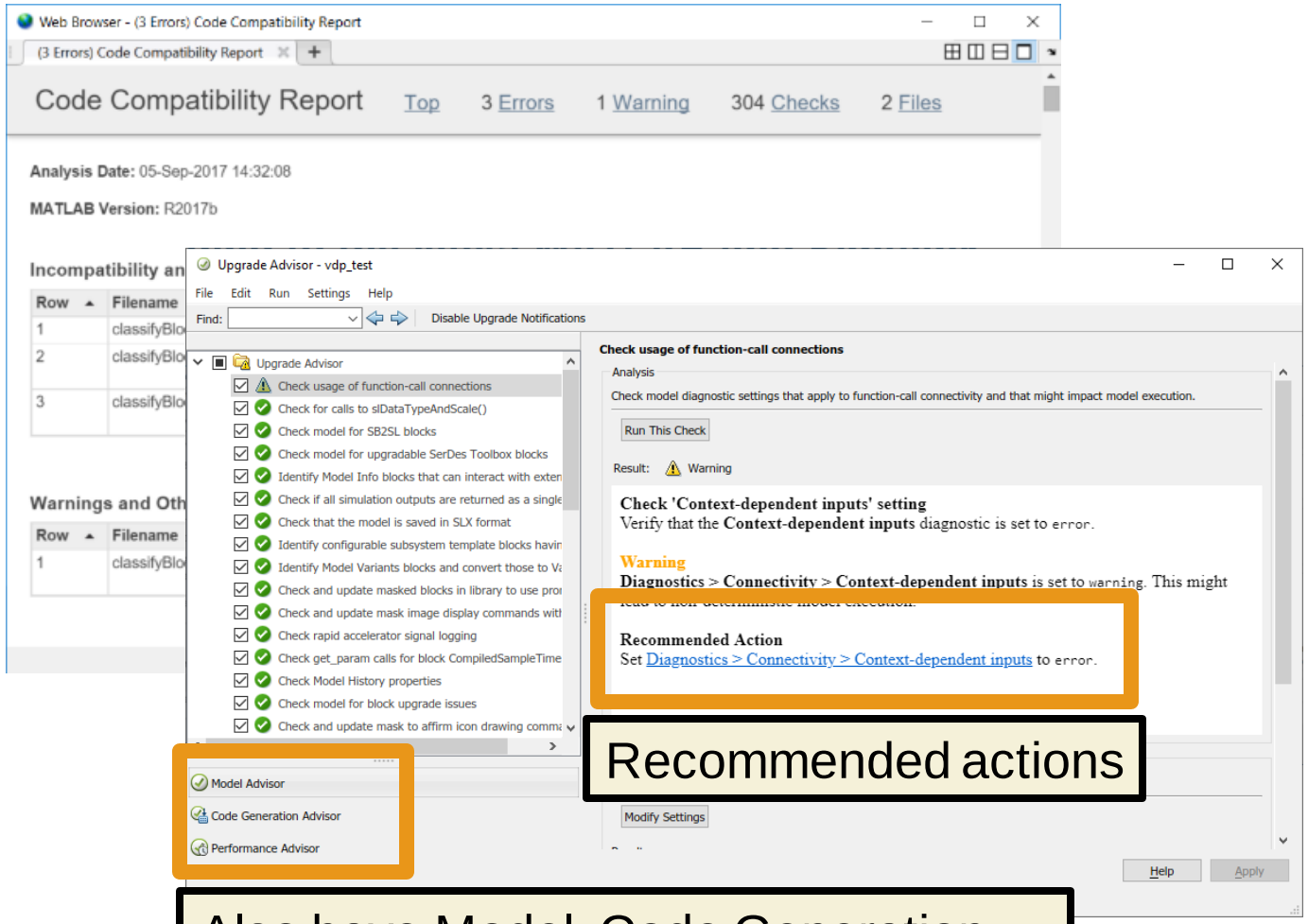
Developing robust software systems in MATLAB and Simulink

- Writing better and faster code
- Reduce complexity with refactoring
- Integrating with other languages and tools
- Sharing and reuse

Upgrading to the Latest Version of MATLAB and Simulink

- Code Compatibility Report

- Upgrade Advisor



Simplify Function Argument Validation and Error Checking

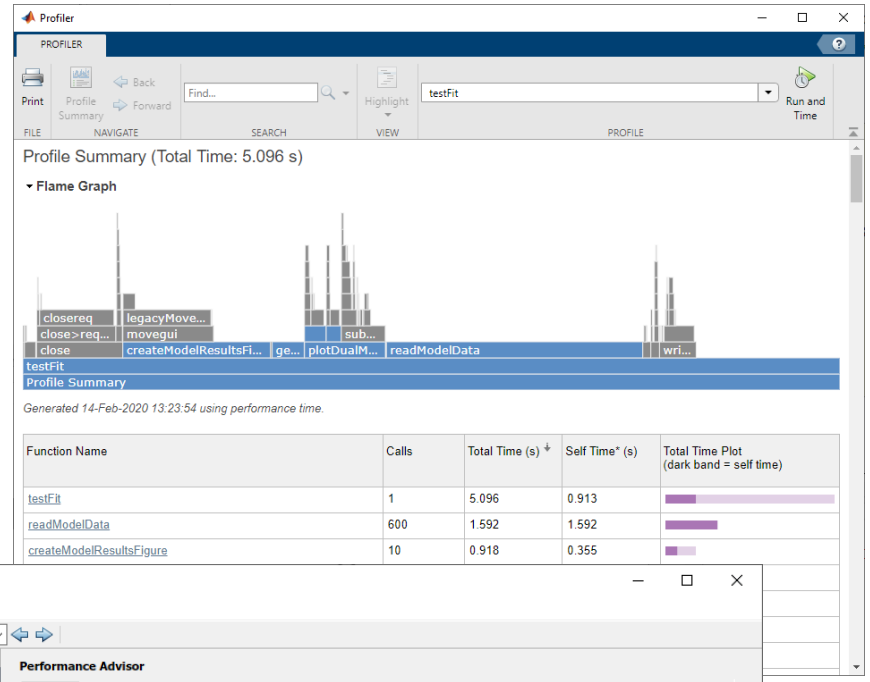
```
% Error check required input arguments
if nargin < 1
    error("rectangle requires width and height values");
elseif ~isnumeric(width) || ~isscalar(width)
    error("width must be a scalar numeric value")
elseif ~isnumeric(height) || ~isscalar(height)
    error("height must be a scalar numeric value")
end

% Process optional inputs xStart and yStart
xStart = 0;
if nargin > 2 && isnumeric(varargin{1}) && isscalar(varargin{1})
    xStart = varargin{1};
end
yStart = 0;
if nargin > 3 && isnumeric(varargin{2}) && isscalar(varargin{2})
    yStart = varargin{2};
end
```

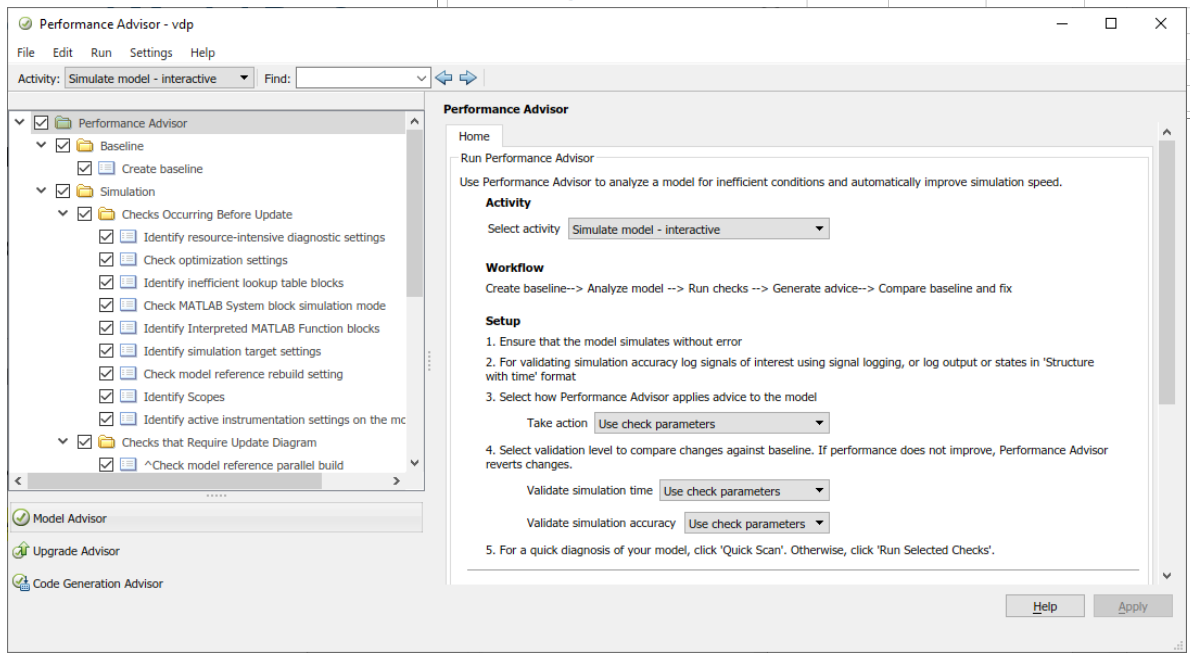
```
arguments
    width  (1,1) double {mustBeNumeric}
    height (1,1) double {mustBeNumeric}
    xStart (1,1) double {mustBeNumeric} = 0; % optional
    yStart (1,1) double {mustBeNumeric} = 0; % optional
end
```

Improving Code and Model Performance

- MATLAB Profiler
  - Flame graph to highlight the largest code bottlenecks
  - Total number of function calls
  - Time per function call
  - Statement coverage of code

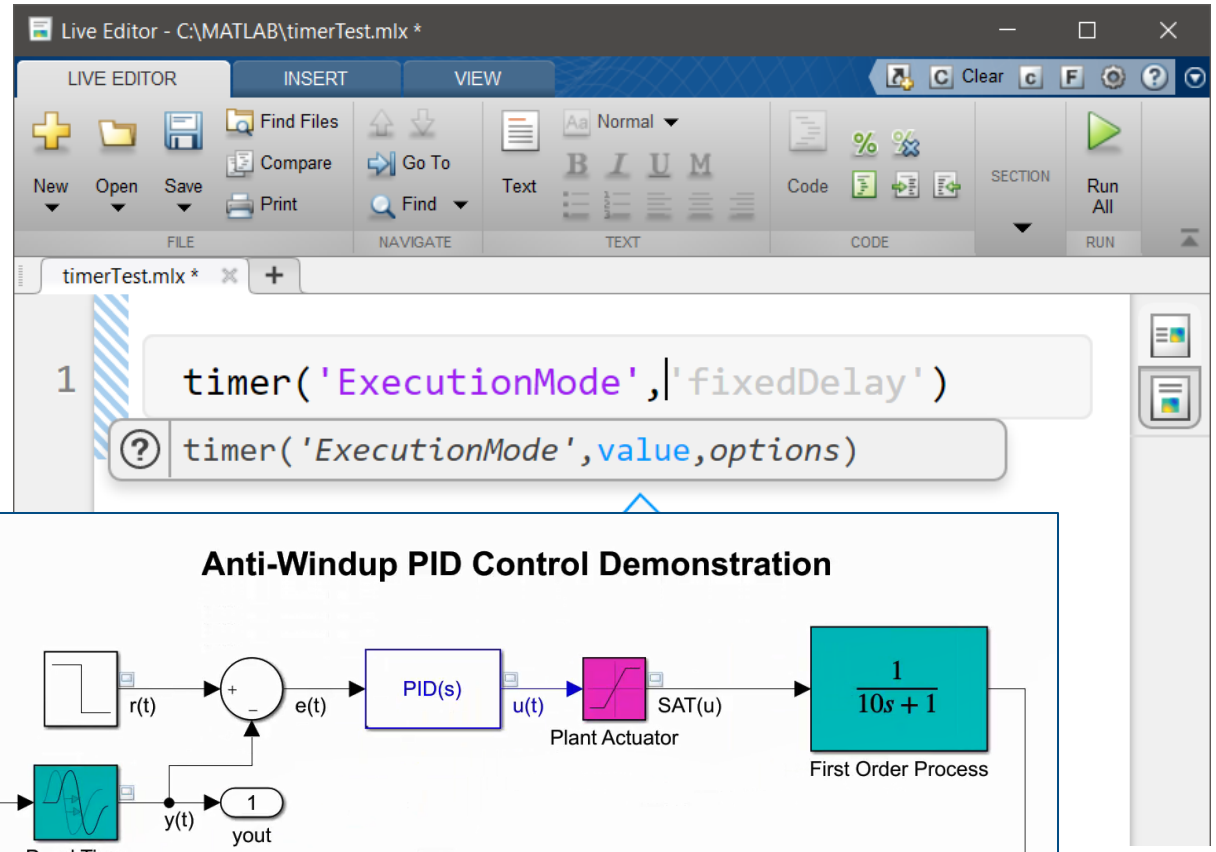
- Performance Advisor in Simulink
  - Create baselines to compare against
  - Review recommendations and automatically apply changes



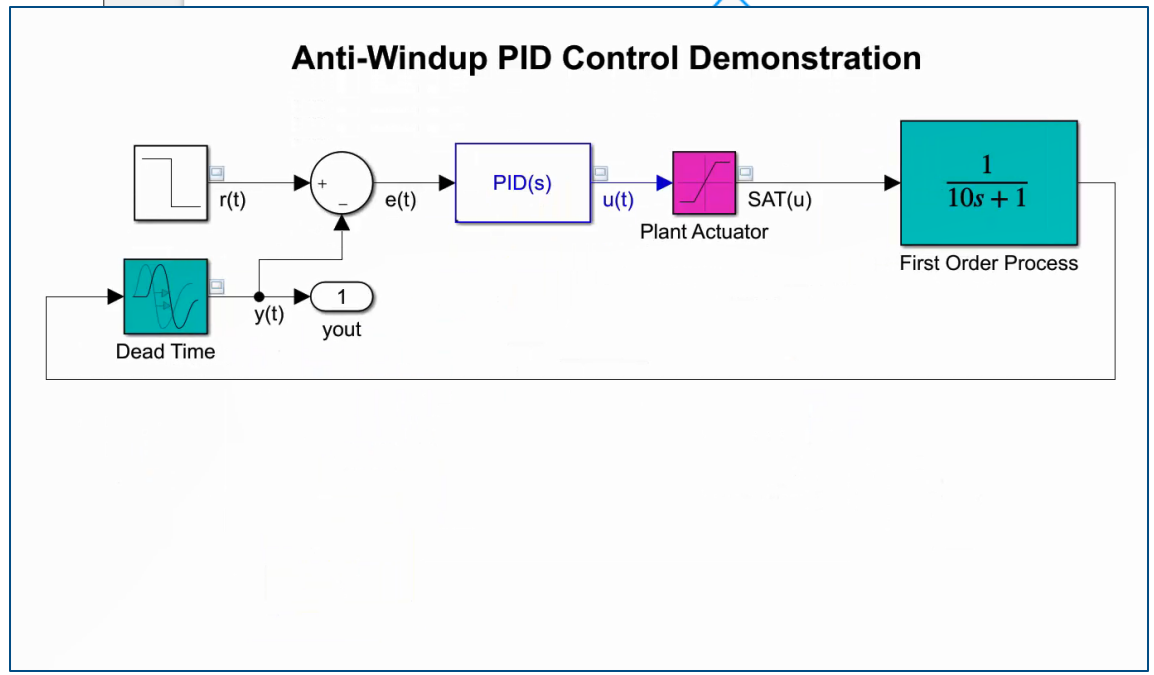


Speed up Your Development

- Context-aware coding guides
  - Automatically suggest functions, variables, files, and Name-Value pairs



- Model layout tools
  - Automatically clean up messy and complex models



Quickly and Safely Refactoring – MATLAB Code

- Break down large, complex codes and models into reusable and easier to maintain components

Calculate my answer:

```
3  z1 = x+y;
4  z2 = x-y;
5  z3 = y-x;
6  z4 = x*y;
7  zSum = z1 + z2 + z3 + z4;
```

Display answers of interest:

```
8  disp(z3)
9  disp(zSum)
```

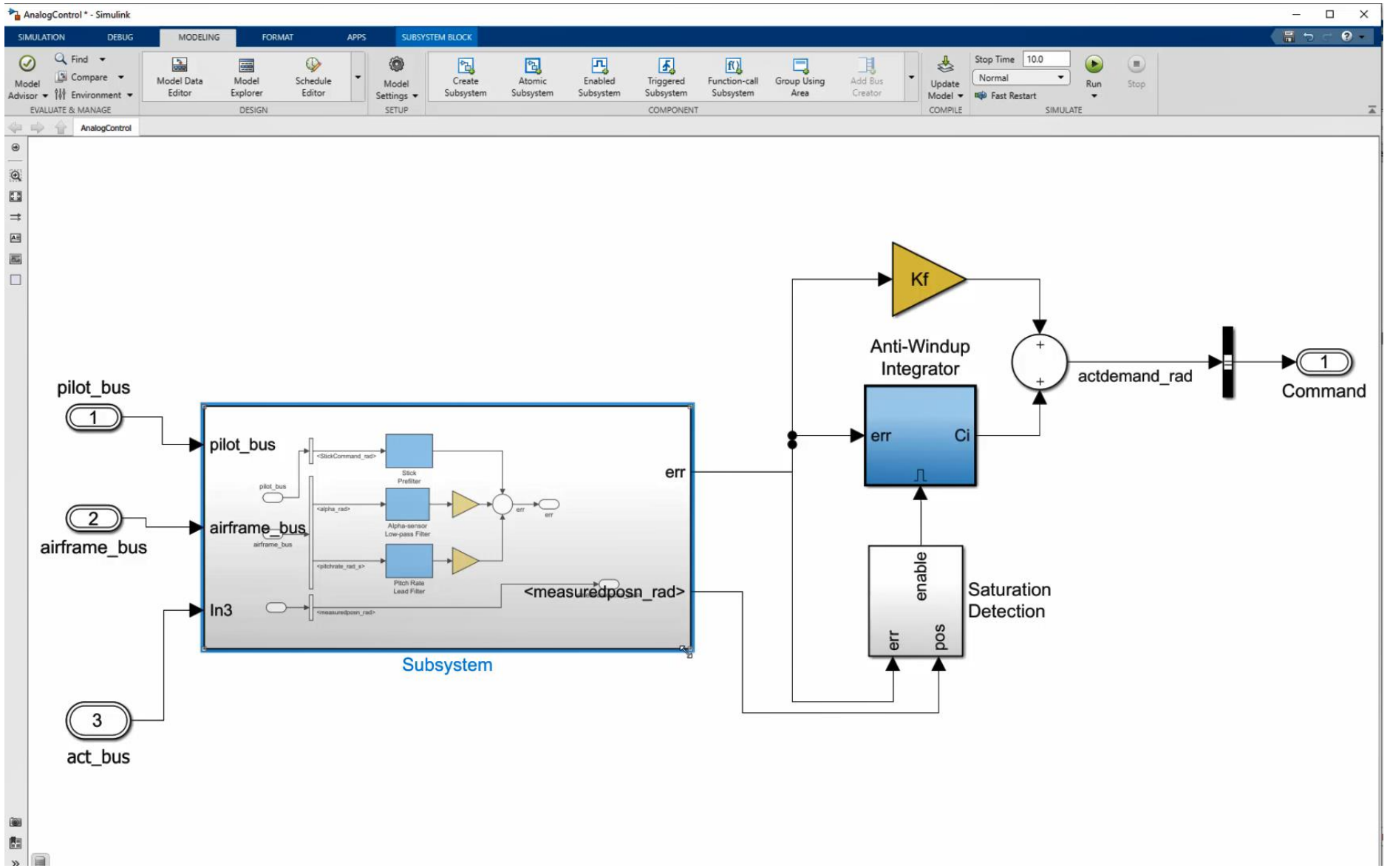
Evaluate Selection in Command Window	F9
Open Selection	Ctrl+D
Help on Selection	F1
Copy Output	
Copy All Output	
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Comment	Ctrl+R
Uncomment	Ctrl+T
Convert Between Code and Text	Ctrl+E
Change Case	Ctrl+Shift+A
Smart Indent	Ctrl+I
Convert to Function	
Convert to Local Function	
Insert Section Break	Ctrl+Alt+Enter
Run Section	Ctrl+Enter

Live Editor - C:\MATLAB\TestFolder\myMathFunction.mlx

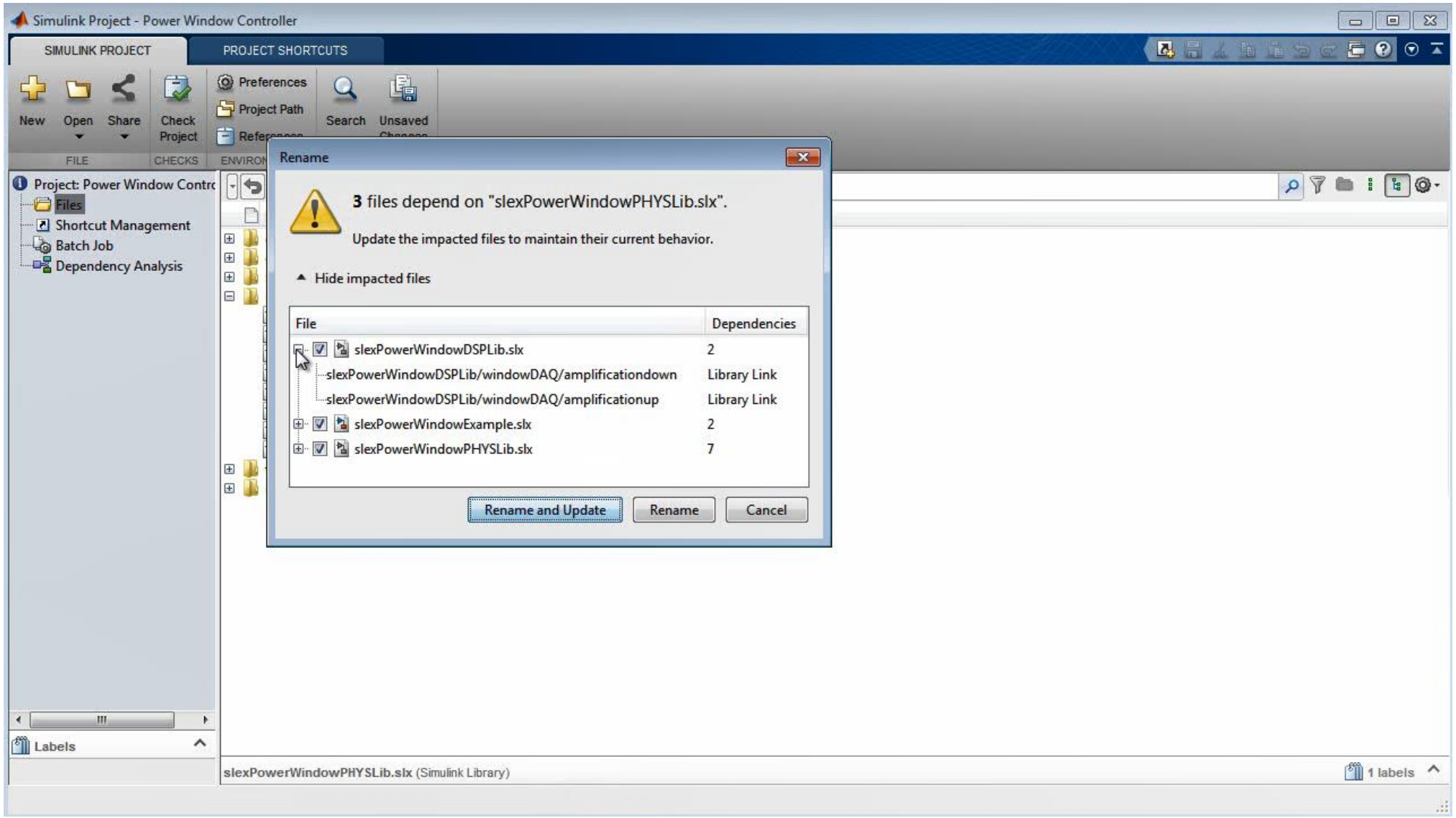
MainScript.mlx * | myMathFunction.mlx

```
1  function [z3, zSum] = myMathFunction (x, y)
2  z1 = x+y;
3  z2 = x-y;
4  z3 = y-x;
5  z4 = x*y;
6  zSum = z1 + z2 + z3 + z4;
7  end
```

Quickly and Safely Refactoring – Simulink Models

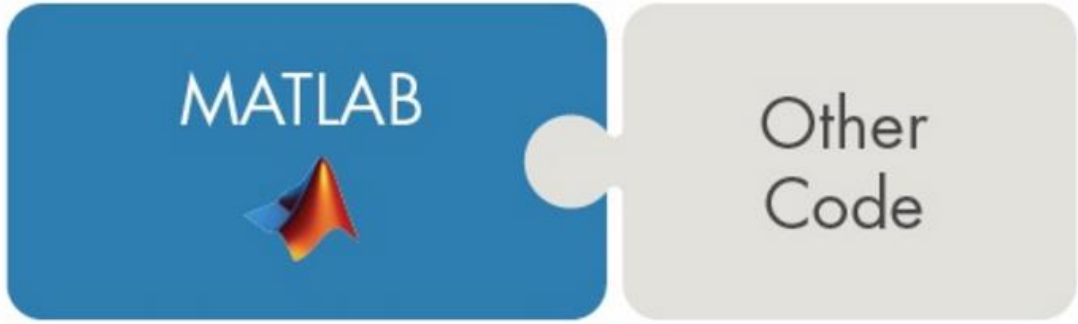


Quickly and Safely Refactoring – Function and Model Names



Integrating with other languages

Calling Libraries Written in Another Language



- Java
- Python
- C/C++
- Fortran
- COM components and ActiveX® controls
- RESTful, HTTP, and WSDL web services

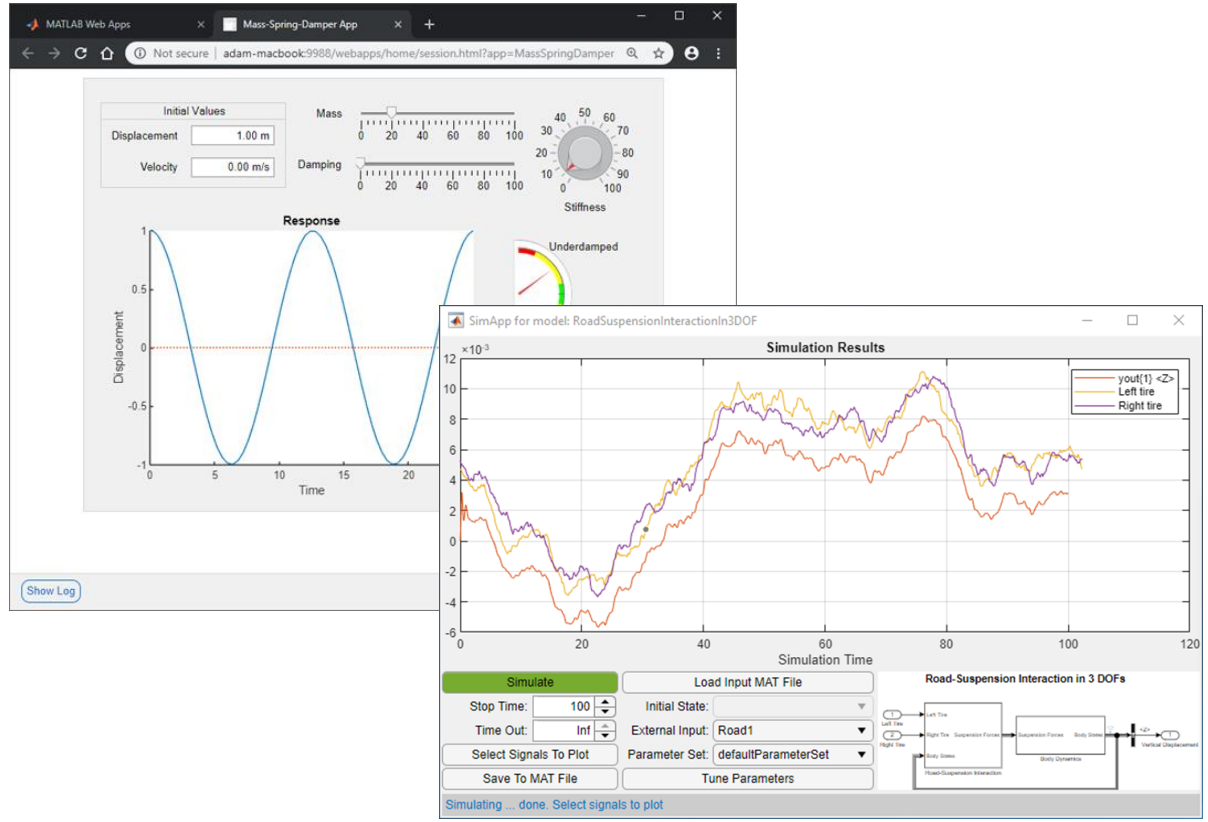
Calling MATLAB from Another Language

- Java
- Python
- C/C++
- Fortran
- COM Automation server

Integrating with other languages

Sharing your work

- Co-authors and development teams
  - Projects
- End-user with MATLAB and Simulink
  - Toolbox or App
- End-user without MATLAB and Simulink
  - Standalone and web applications
  - Language-specific libraries
  - Generated standalone code
  - Microservice APIs



MATLAB Compiler, Simulink Compiler

MATLAB Compiler SDK

Embedded Coder, HDL Coder, PLC Coder, GPU Coder, …

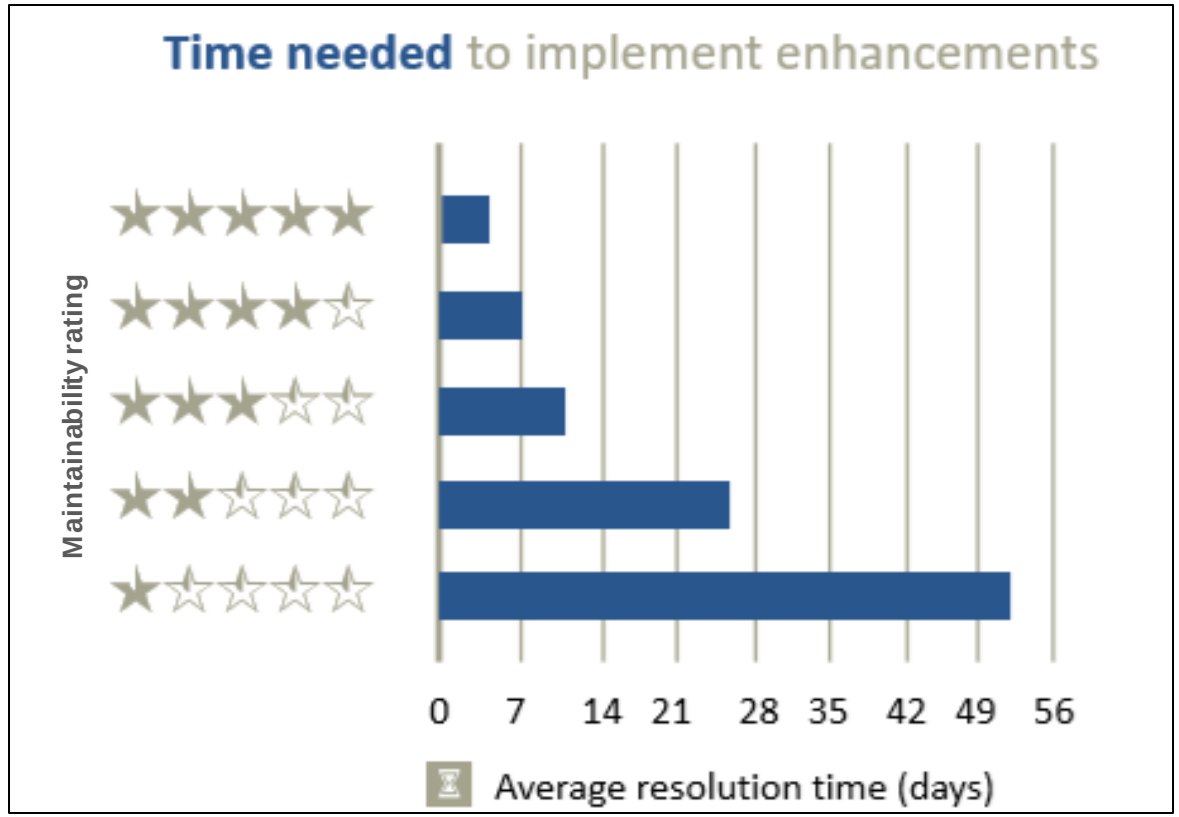
MATLAB Production Server

Agenda

	Setting up your development environment
	Managing team workflows
	Developing better code and models
	Testing and verification

Software Maintenance – The hidden cost of development

- How do you ensure code and models don't break over time?
- How do you keep new features from breaking existing features?
- How do you maintain confidence that your system is working as expected?
- How do you ensure that your software is future-proof?

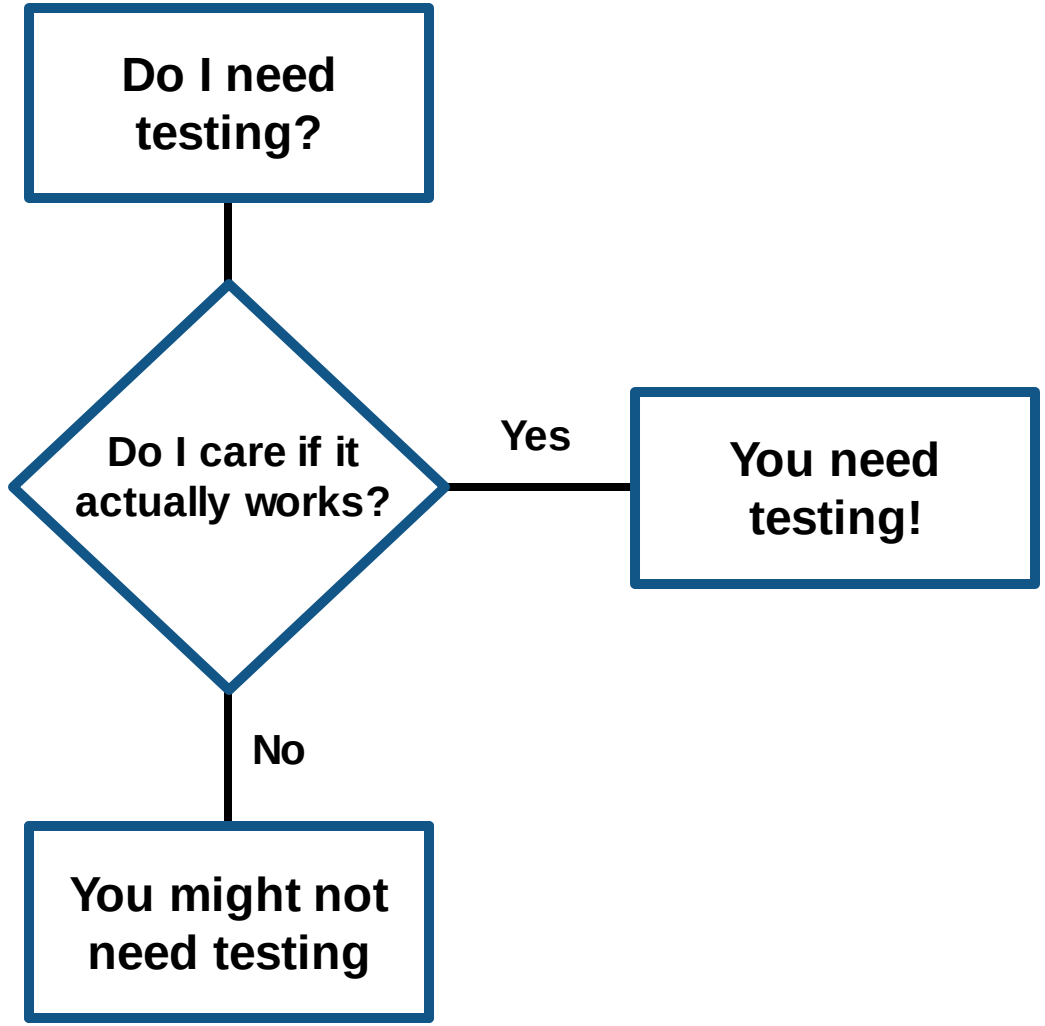


Journal paper: "*Faster issue resolution with higher technical quality of software*", Software Quality Journal, 2011

Test early, test often, test automatically

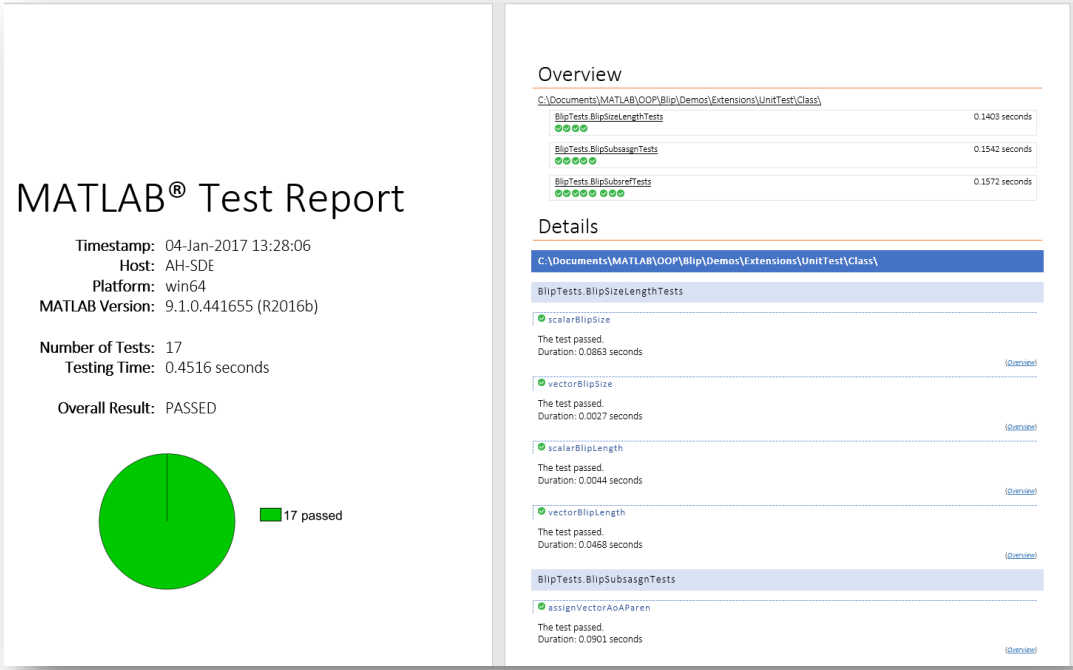
- Reduce risk of software breaking
- Catch problems early
- Improve quality
- Document expected behaviour

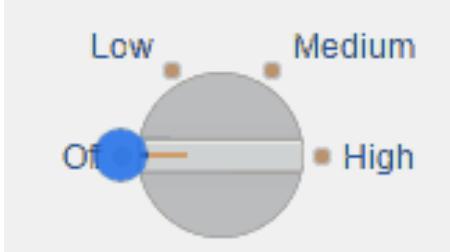
Credit: http://geek-and-poke.com/

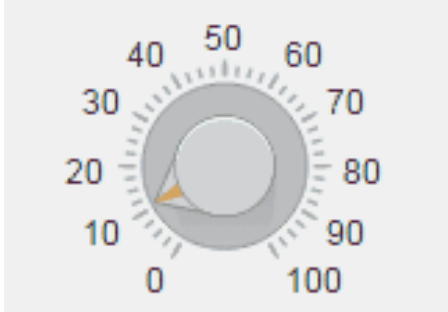


MATLAB Testing Frameworks

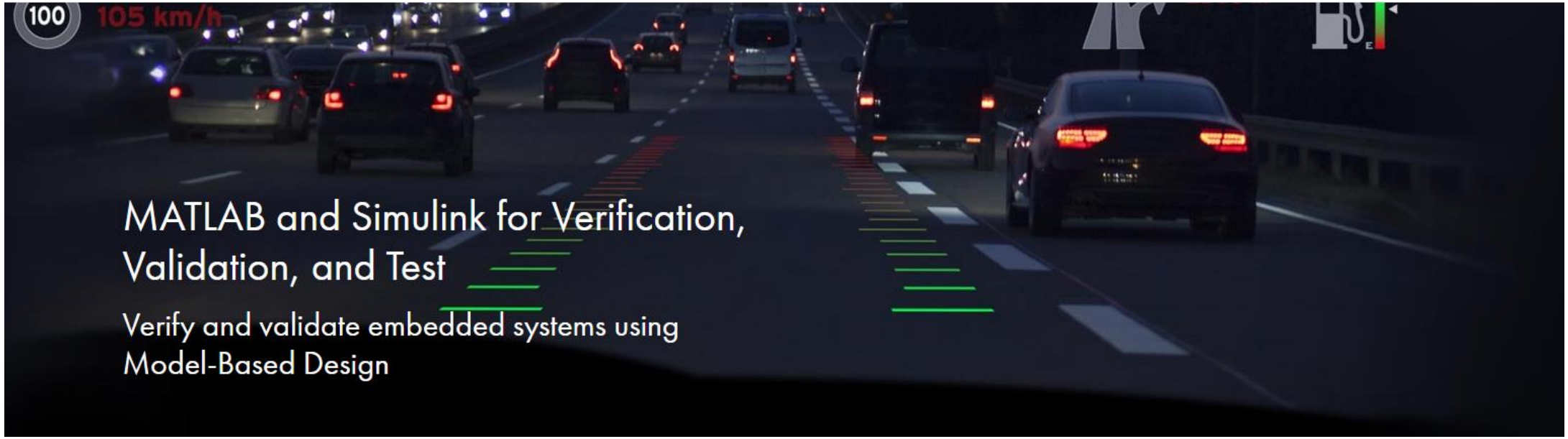
- MATLAB Unit Testing Framework
- Performance Testing Framework
- Mocking Framework
- App Testing Framework







Verification and Validation in Simulink



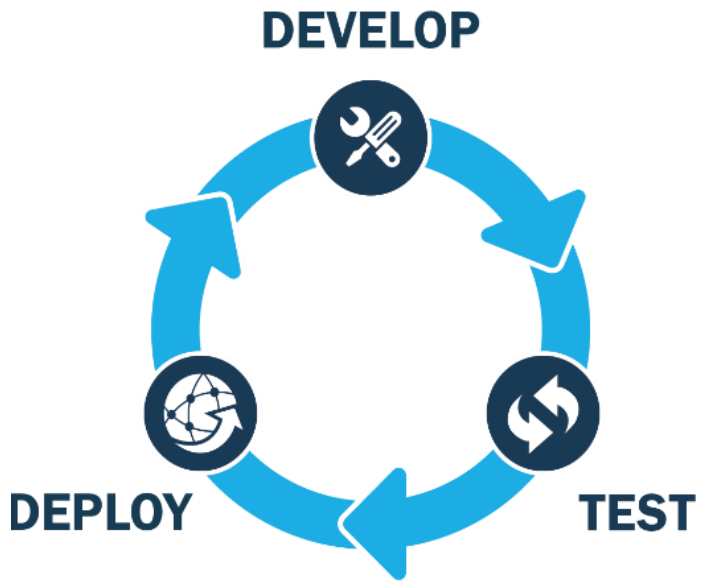
- **Trace requirements** to architecture, design, tests, and code
- **Verify your design** meets requirements and is free of critical run-time errors
- **Check compliance** and measure quality of models and code
- **Generate test cases** automatically to increase test coverage
- **Produce reports** and artifacts, and certify to standards (such as DO-178 and ISO 26262).

Automated Testing with Continuous Integration (CI)

- A system to automate the building, testing, integration, and deployment of code as it is being developed and maintained

- Popular CI systems: Jenkins, Travis, CircleCI , Azure DevOps, and others…

- Benefits:
  - Detect integration bugs early
  - Allow you to stop bugs from being accepted
  - Track and report testing history
  - Flexible testing schedules and triggers

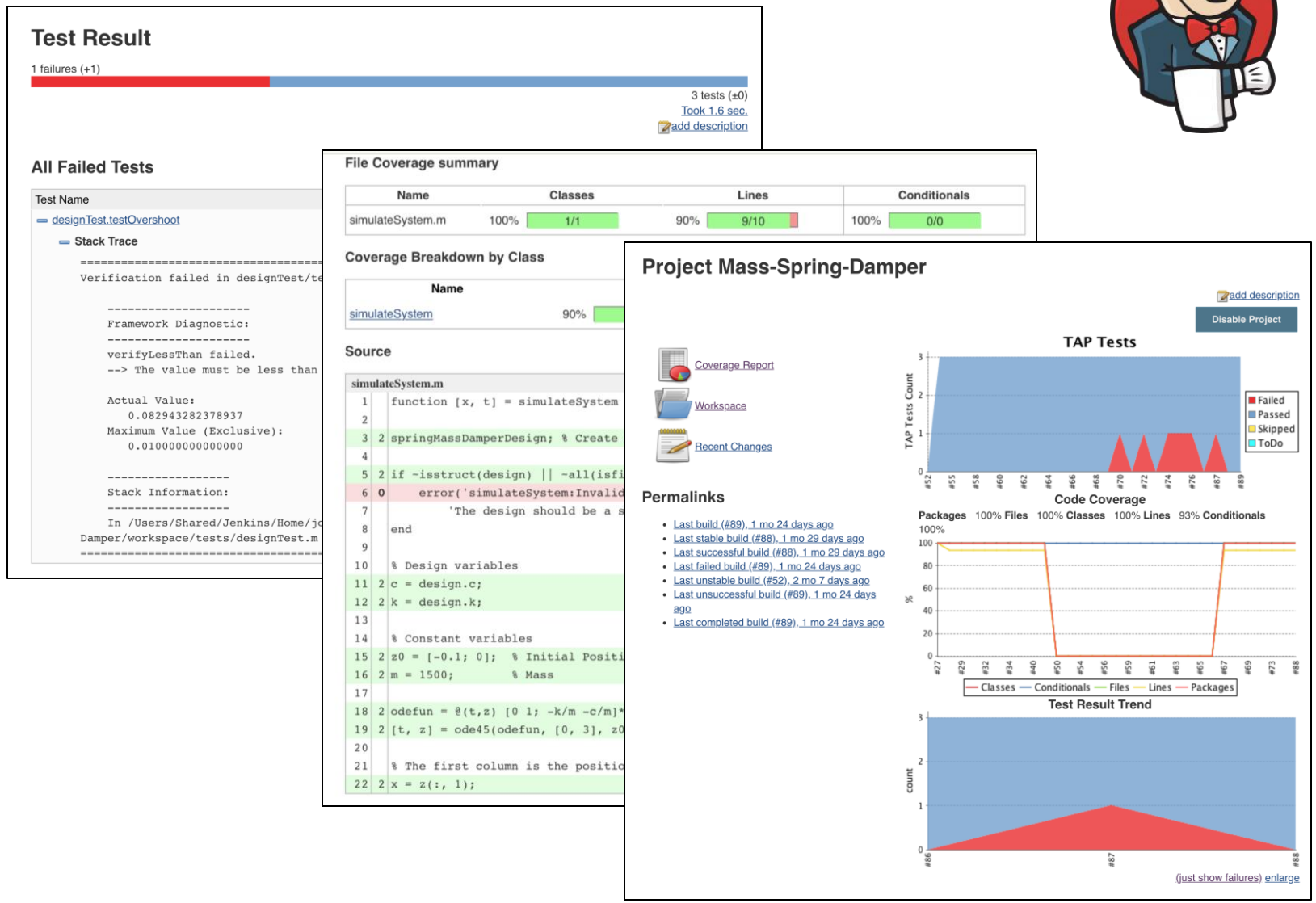


MATLAB Plugin for Jenkins

- Install MATLAB Plugin for Jenkins directly from the Jenkins Plugin Manager
- Easily connect and configure MATLAB with Jenkins
- Schedule automatic code and model testing
  - MATLAB Unit Test Framework
  - Simulink Test

Testing Reports in Jenkins

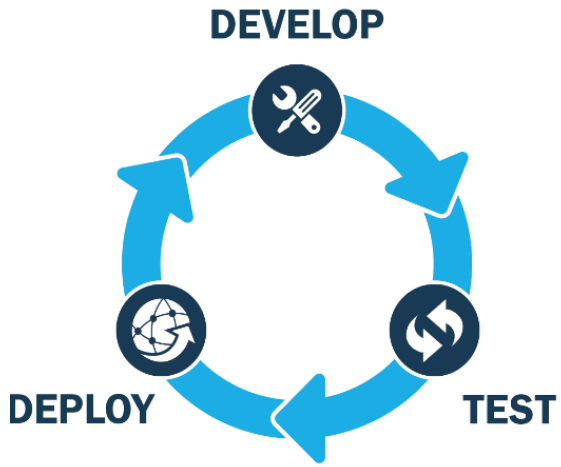
- View testing results
- View code coverage
- View testing reports



Summary

Summary

- MATLAB and Simulink can take you all the way from idea to production
- Save time and effort with good software and modeling practices
- Projects bulletproof your collaborative development workflows



MATLAB and Simulink

are the **easiest** and
most **productive** environments
for **engineers** and **scientists**