MathWorks
AUTOMOTIVE CONFERENCE 2023
India

Accelerating Development for Software-Defined Vehicles Using CI/CD

Nukul Sehgal, MathWorks

Vamshi Kumbham, MathWorks

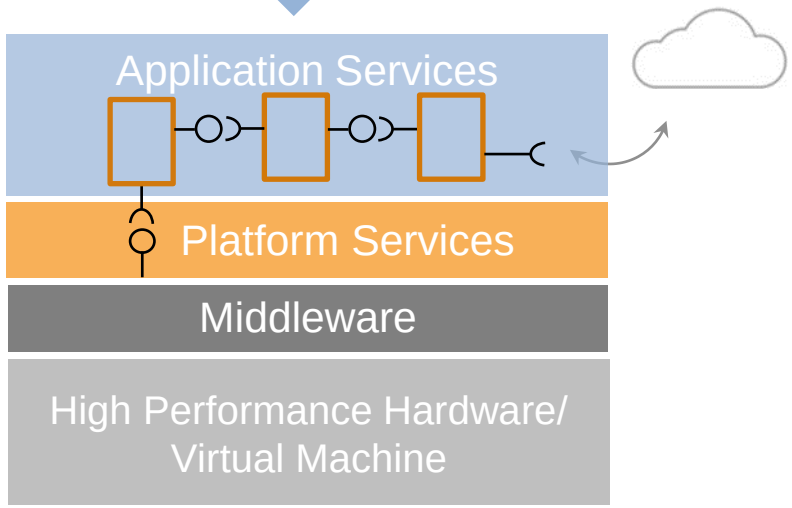
Rajat Arora, MathWorks

MathWorks

Software Defined Vehicles

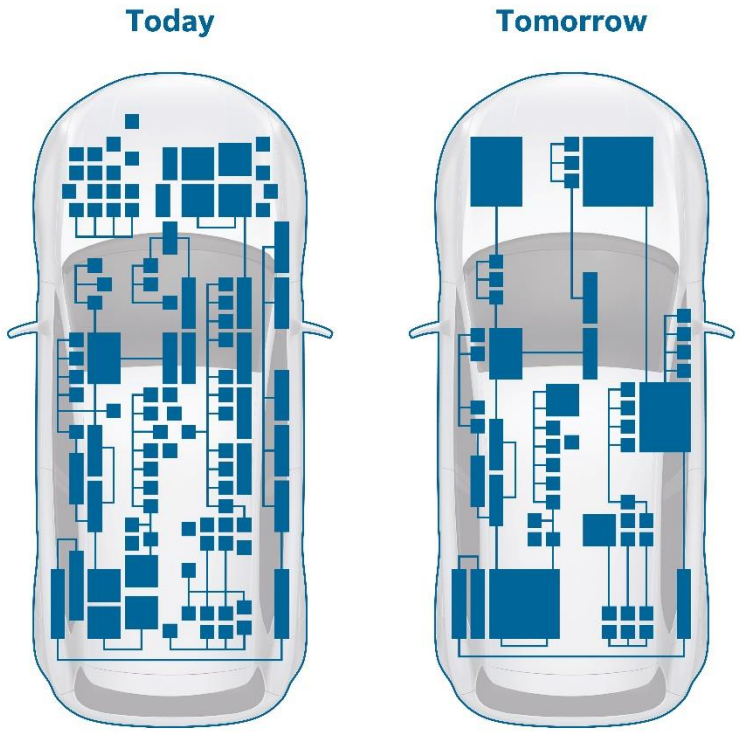
- Automotive industry is embracing Service-Oriented Architectures (SOA) as a new paradigm to design modern applications like Software-Defined Vehicles (SDVs)

SW updates
- Frequent
- Selective
- Over-the-air

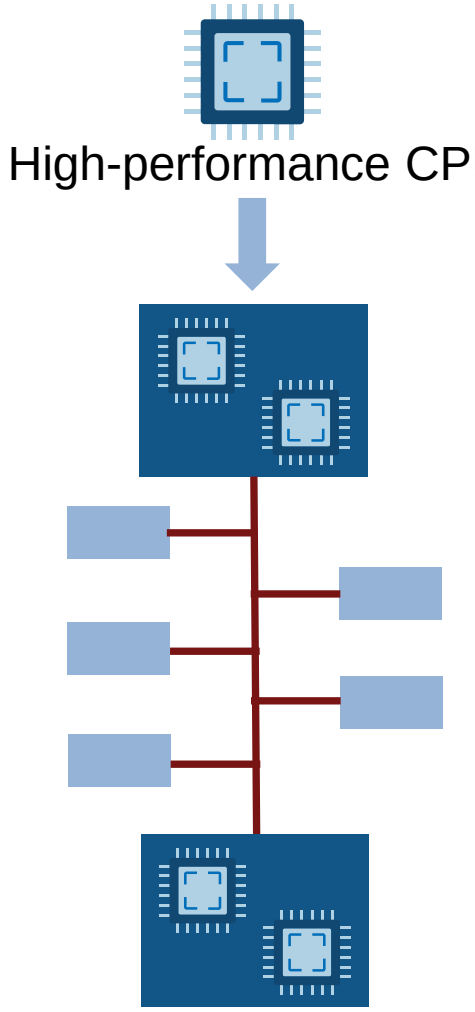


Higher HW abstraction:
Service-oriented architectures

Centralization of computing and SOA



Consolidation and centralization of computing

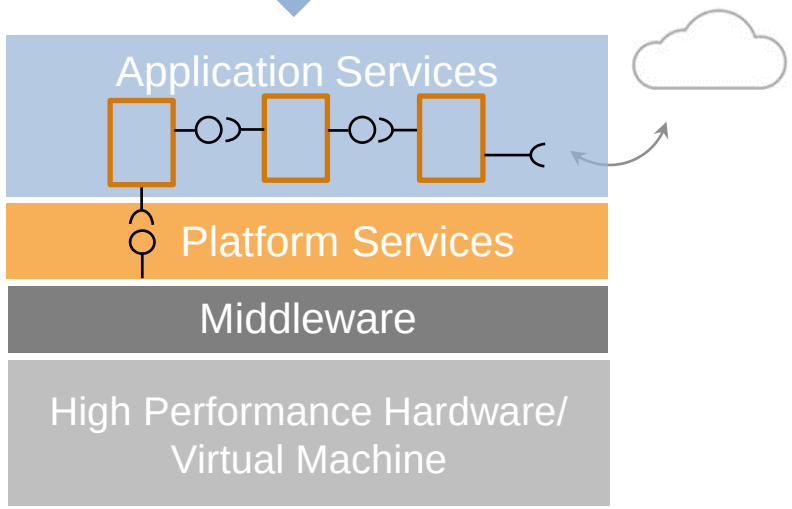


High-performance CPU/GPU

New E/E zonal architectures

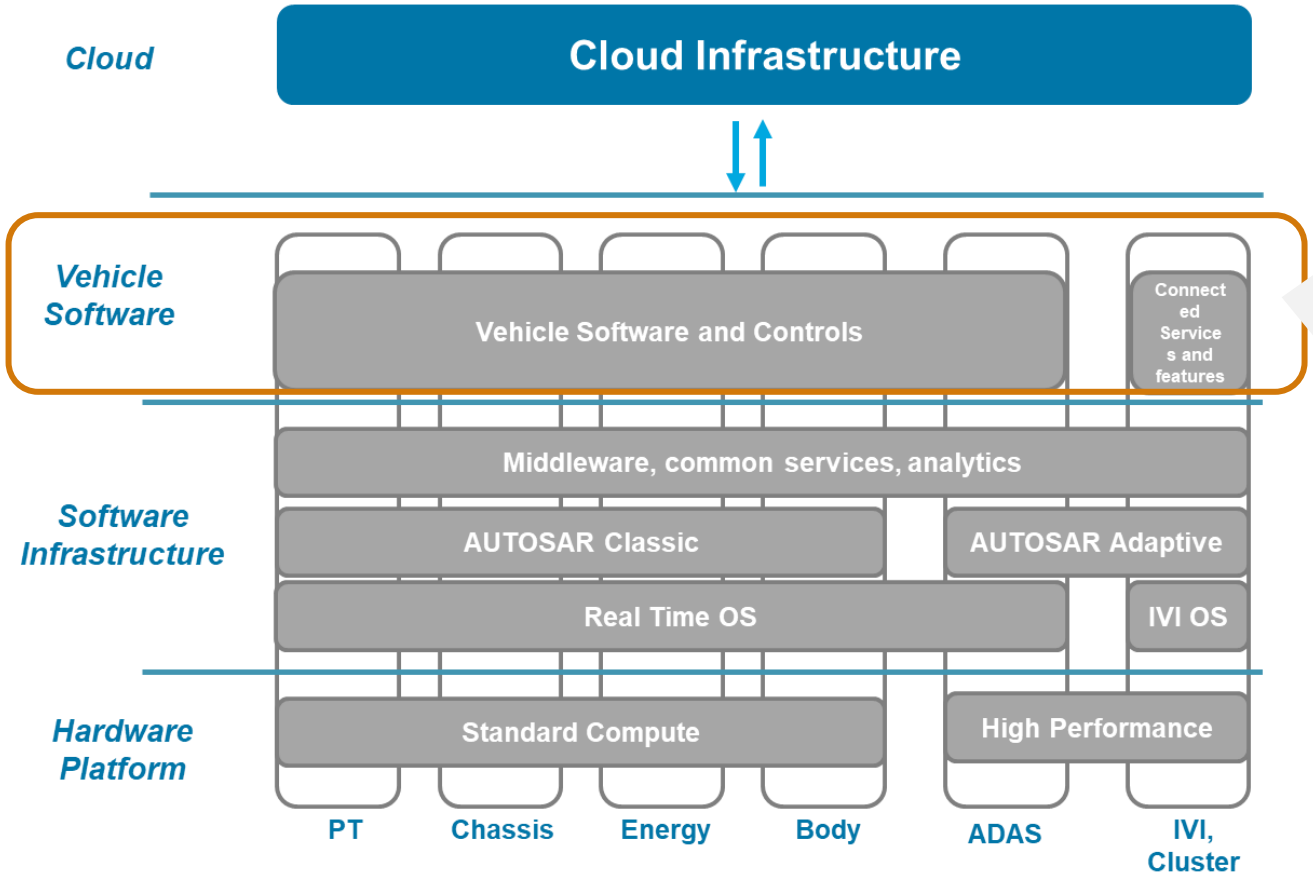
SW updates

- Frequent
- Selective
- Over-the-air



Higher HW abstraction:
Service-oriented architectures

Modeling and Automation for Software Defined Vehicle Applications



Model-Based Design

Modeling

Simulation | Analysis

Fast repeatable tests

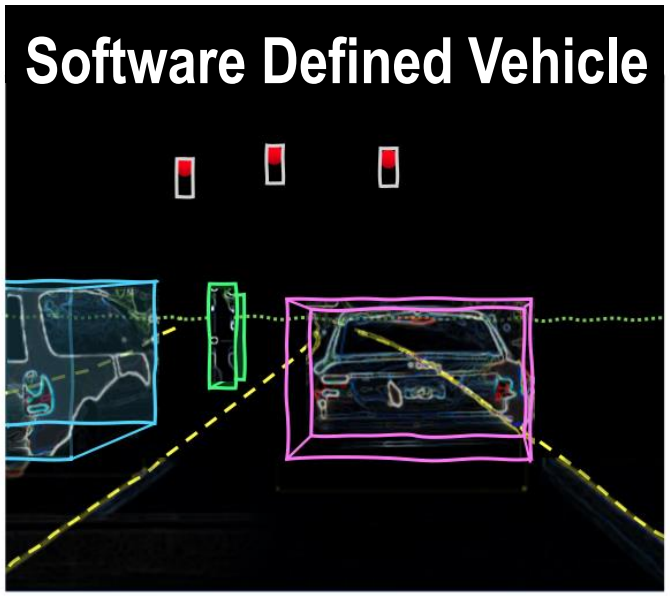
Automation

Coding | Verification

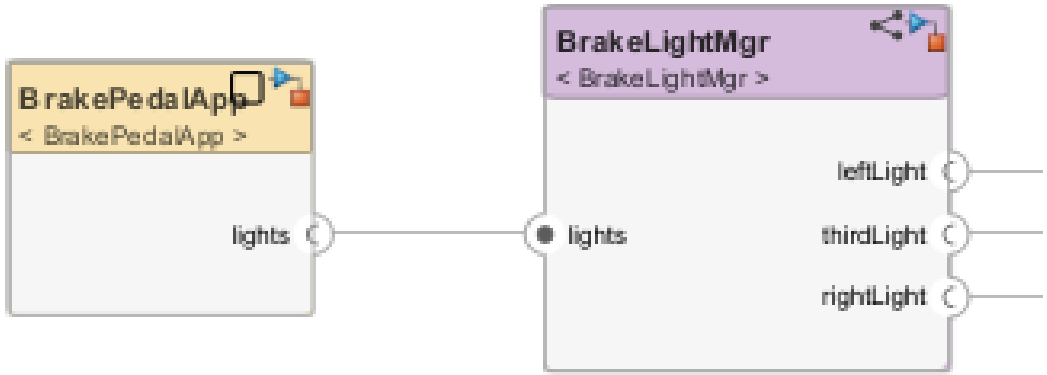
Short agile iteration cycles

Model based design is the Systematic use models throughout your development process

Smart CARS call for smart ways to write code



Simulate with..



Build with..

Abstraction of..

Control | Planning | Perception

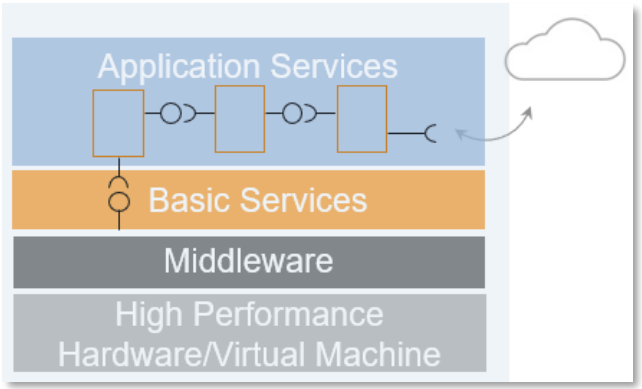
Service-Oriented Architecture

Models that help you work at the right level of abstraction

Agenda

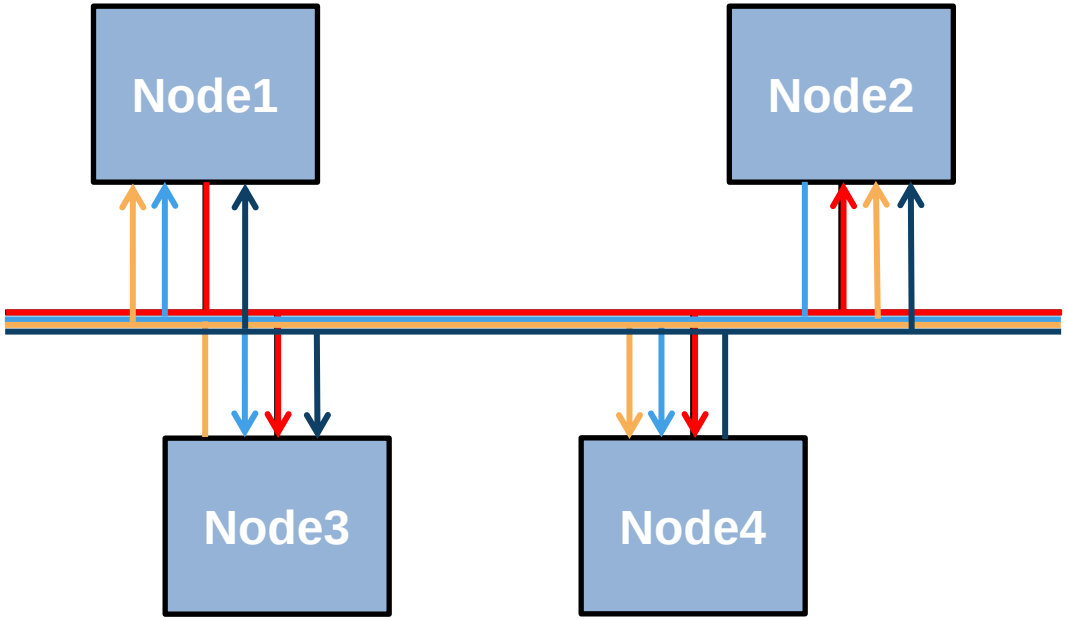
- From Architecture to Deployment
  - Developing SOA Applications
  - Deploying to Linux Targets & Virtual ECUs on Cloud
- Scaling For Production : CI/CD and Model DevOps
  - Case Study : Cruise Control System

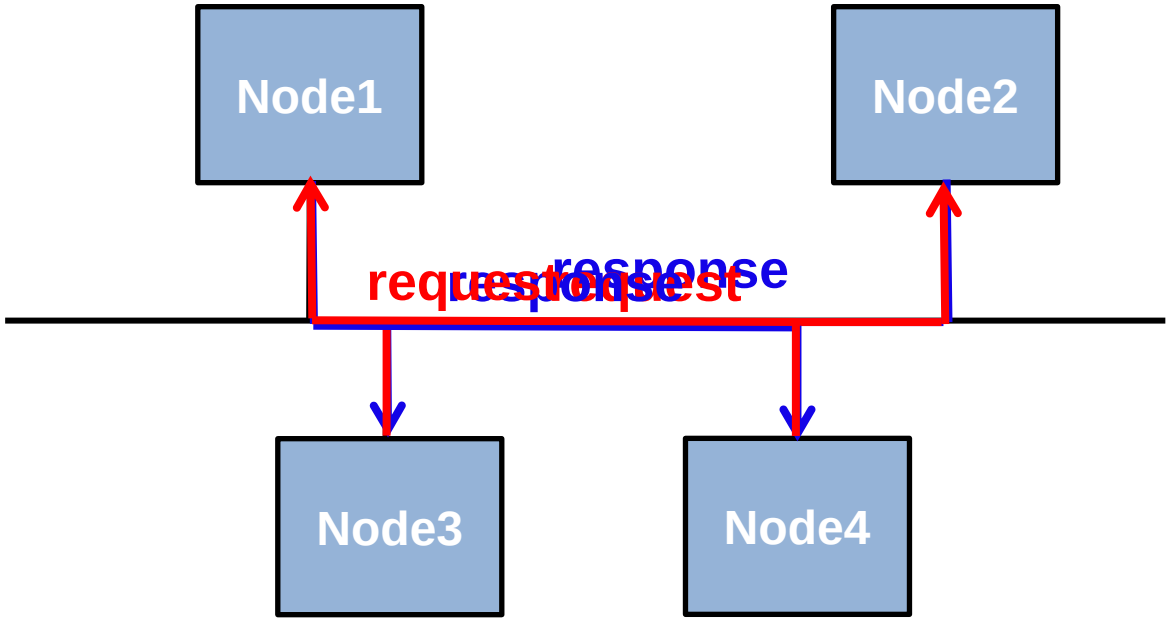
SOA – What's it all about?



- SOA consists of services that communicate across different platforms over messages.
- SOA provides flexibility to add, remove, or update components without impacting the entire, typically large, software system
- SOA is used by multiple industrial standards including:
  - AUTOSAR Adaptive Platform
  - DDS (Data Distribution Services)
  - ROS (Robot Operating System)

SOC (Service Oriented Communication)





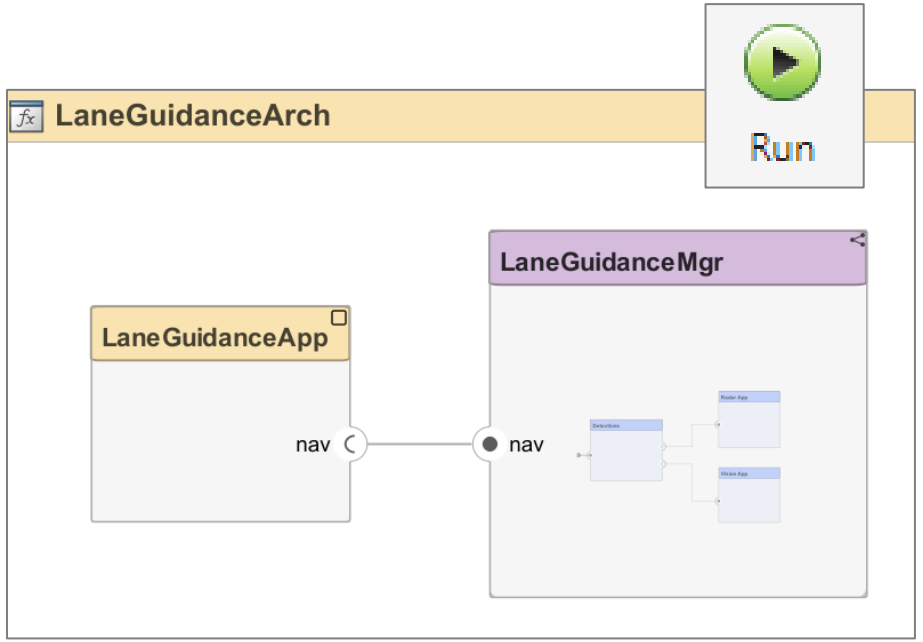
signal-oriented communication

- send data independent of needs
- high bus load
- not efficient

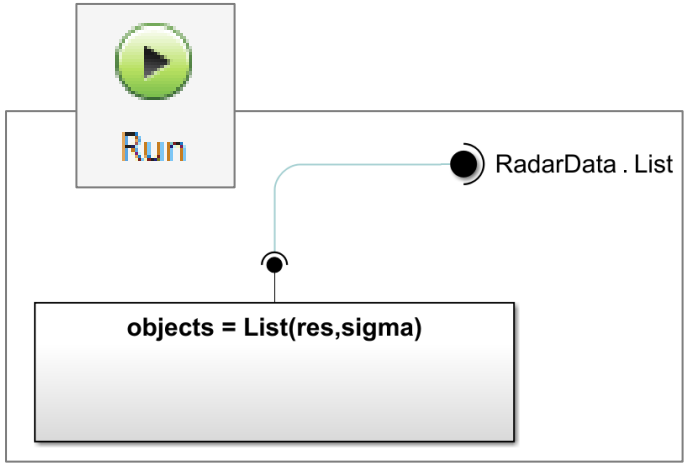
service-oriented communication

- send data dependent of needs
- low bus load
- more efficient

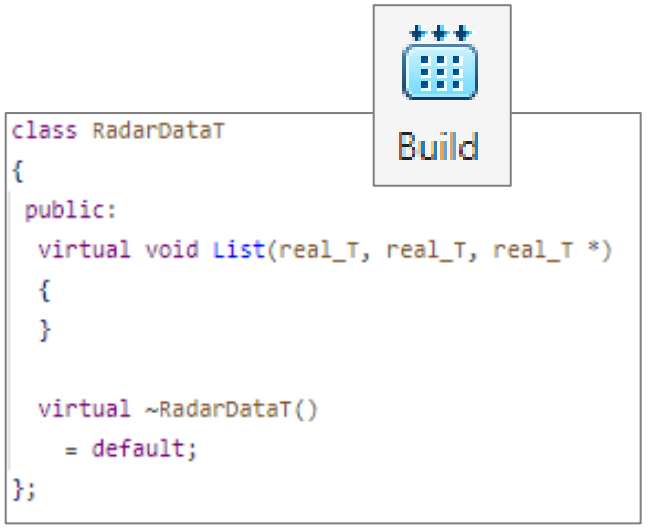
Service-Oriented Architecture (SOA) Design



Describe SOA with
System Composer

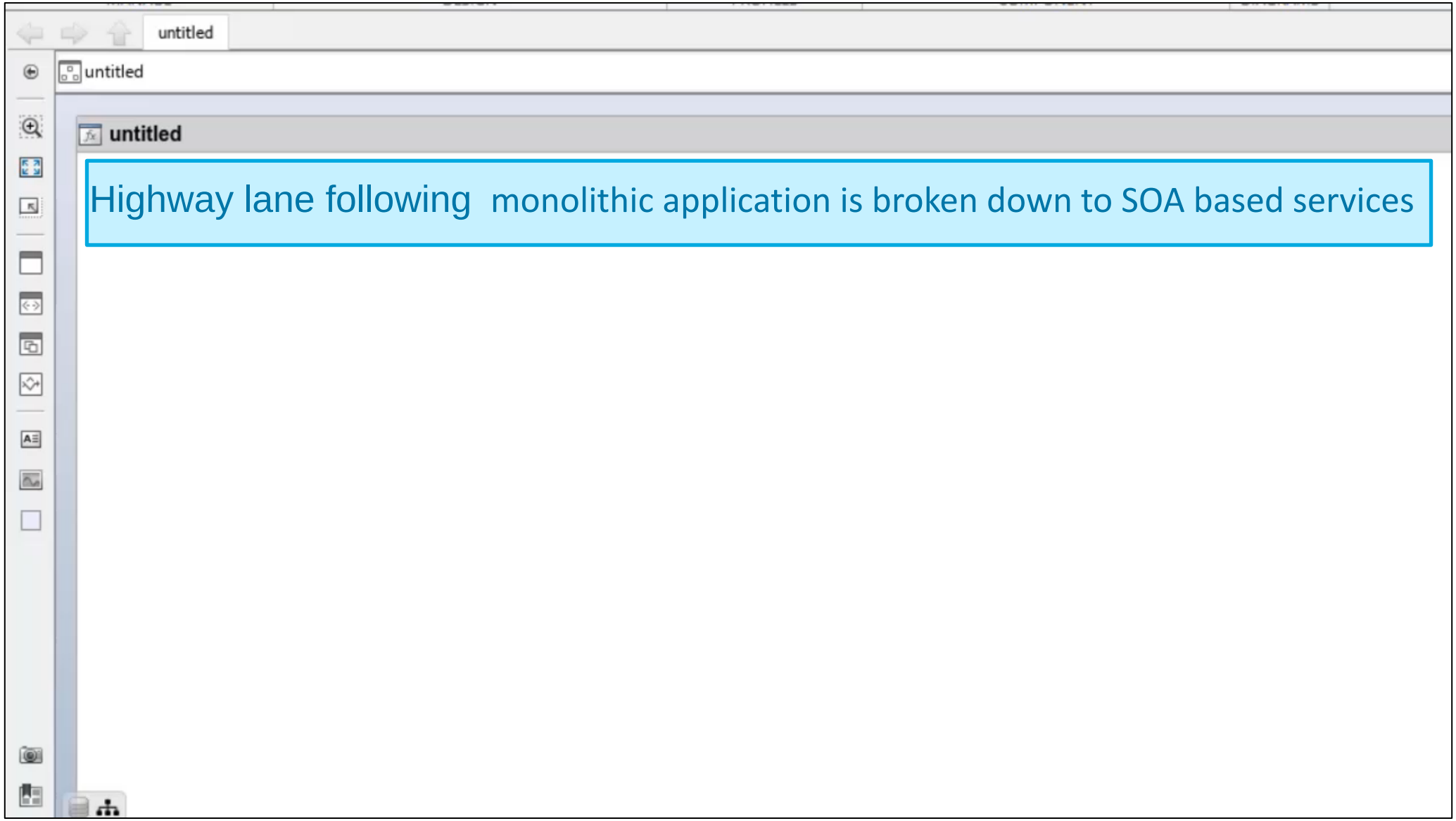


Implement detailed components with
Simulink



Generate code with
Embedded Coder

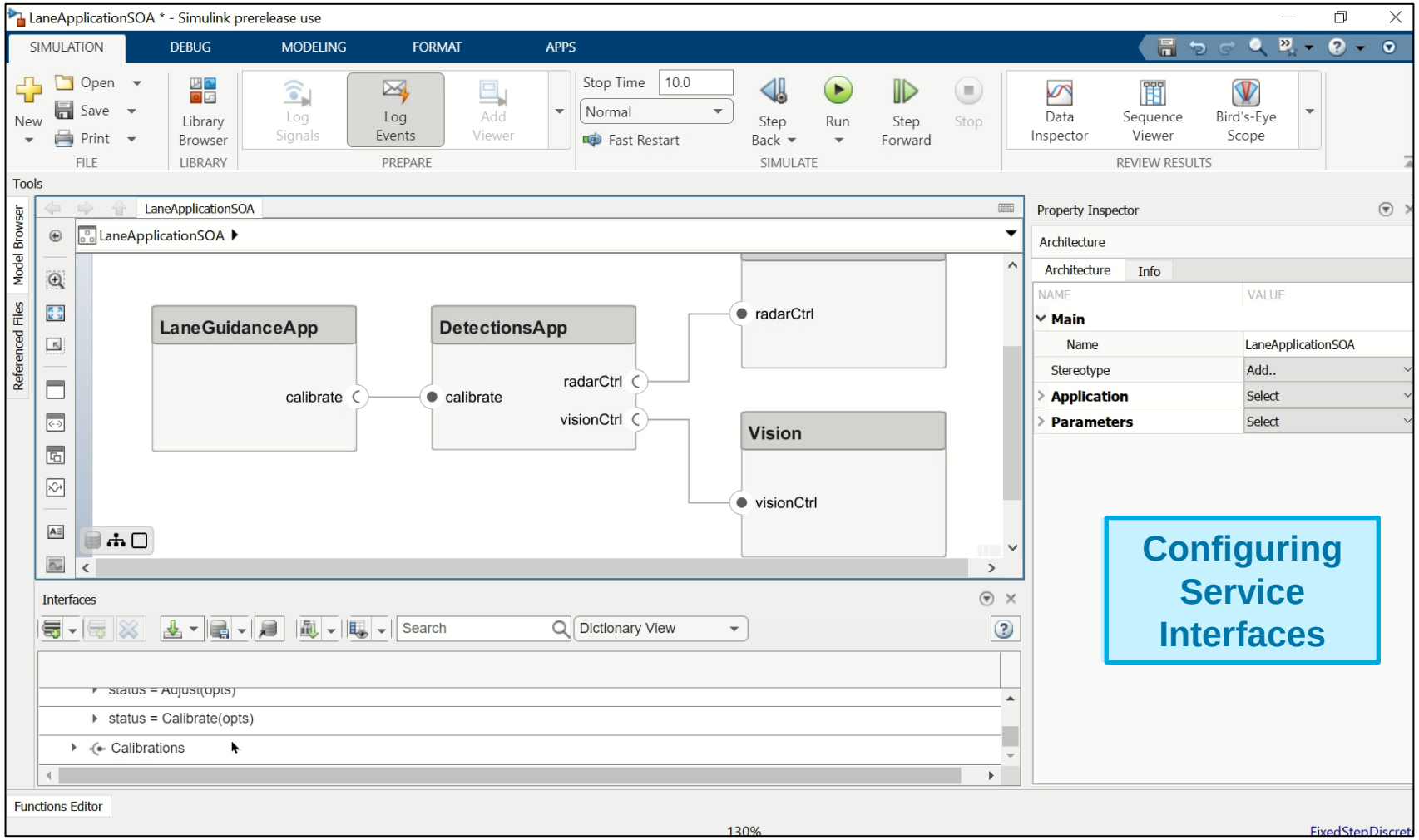
Define Services



Define Service Interfaces

Identify and Analyze Services → Define Services and its interfaces → Define Service Behavior → Implement and deploy Services

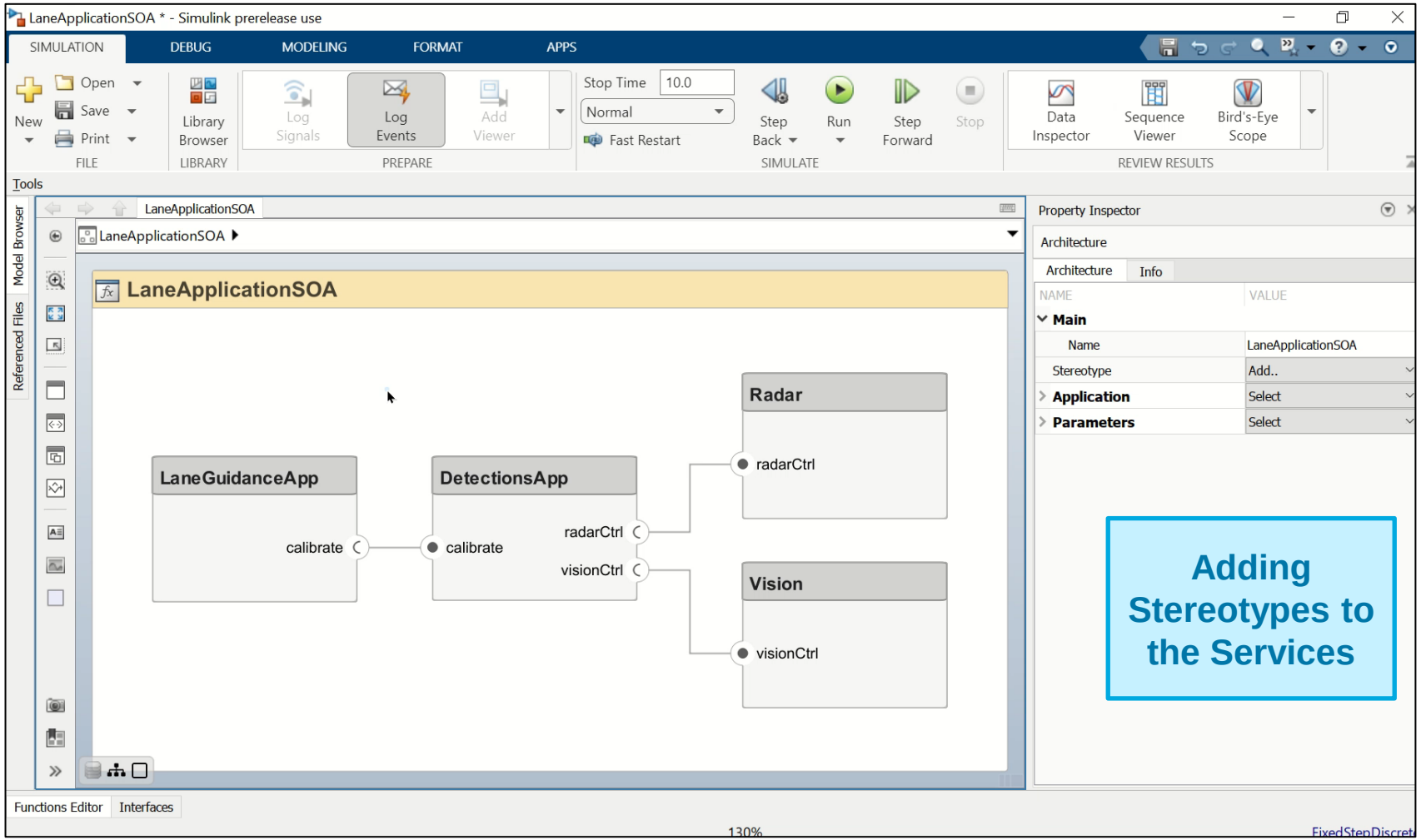
- A service interface is component of service boundary that separates the service from other services and the outside world



Reference - https://chrisrichardson.net/post/refactoring/2020/08/21/ten-principles-for-refactoring-to-microservices.html

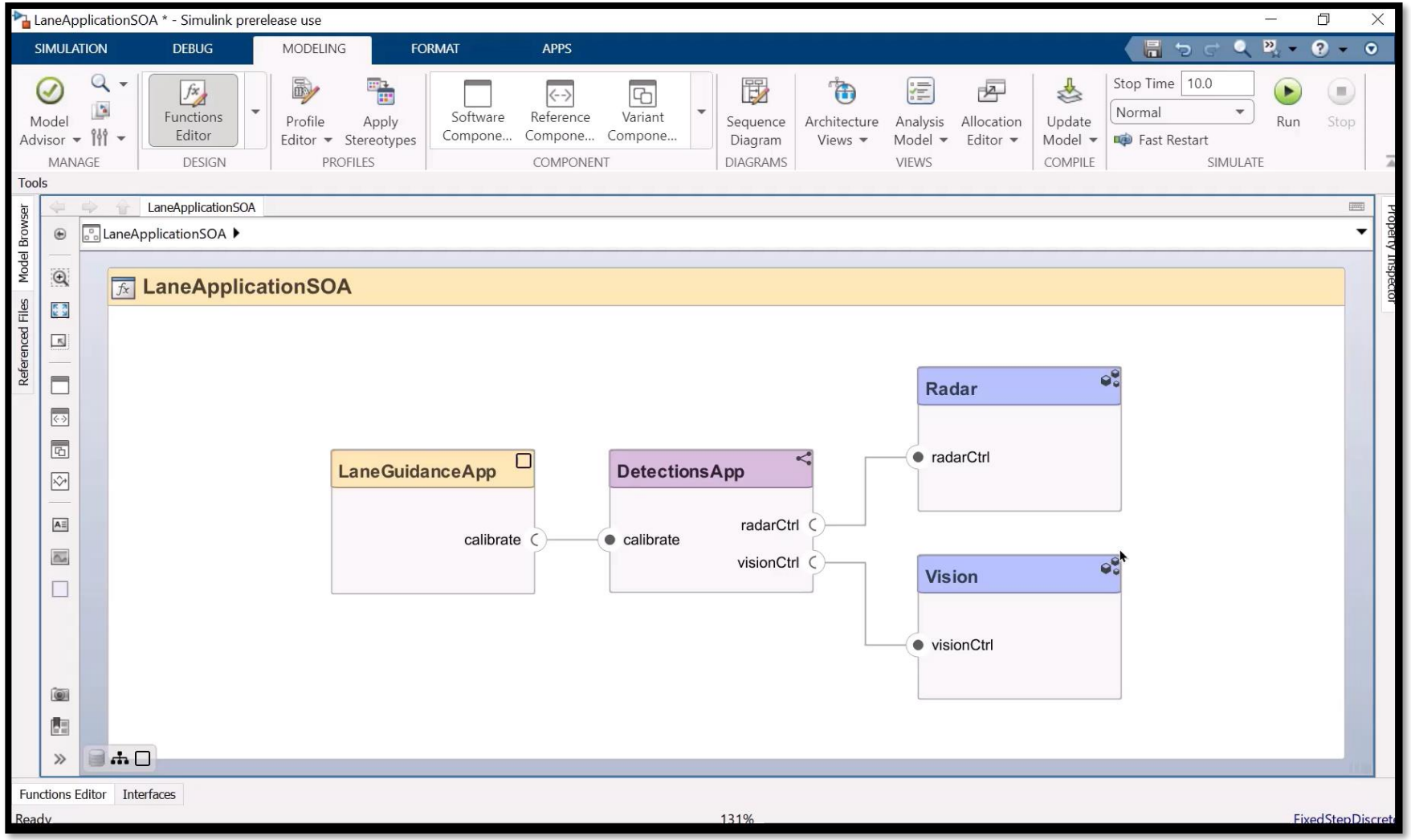
Define Service Stereotypes

- Stereotypes provide service components with a common set of properties.
- It can help in identifying the boundaries of the services and ensures that each service has a clear and distinct responsibility.

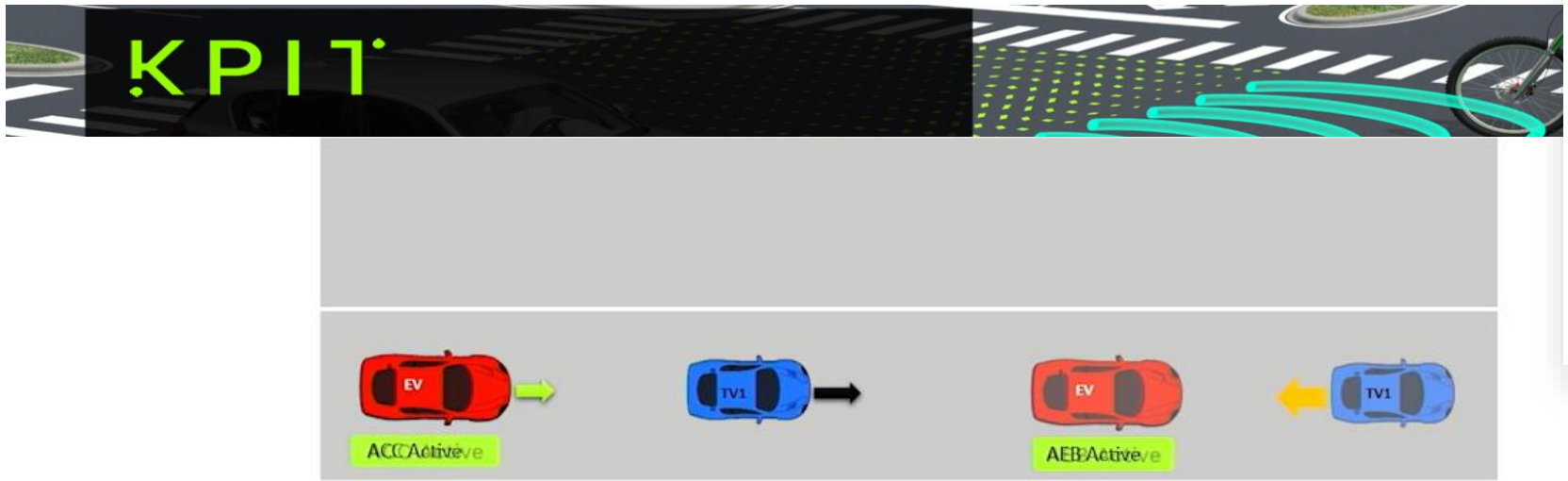


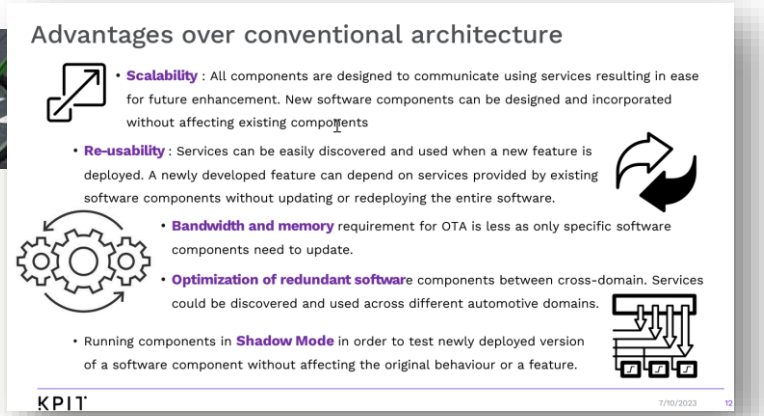
Define Service Behavior

- Each service should have a well-defined inputs, outputs, and behavior.



KPIT- Service-oriented arbitration of ADAS features with Model-Based Design





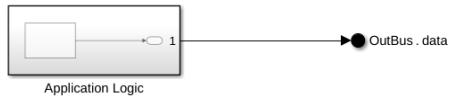
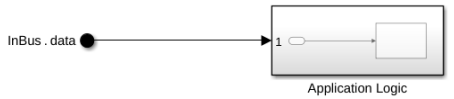
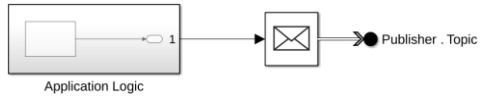
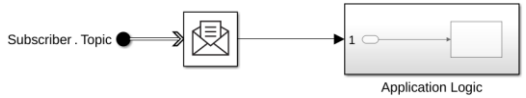
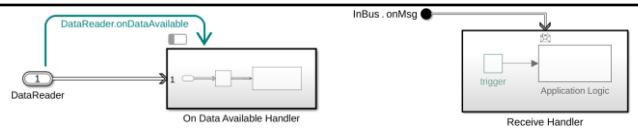
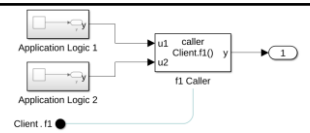
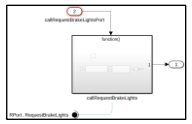
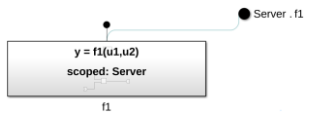
[Link to the talk]

Description: -

- Traffic Vehicle (TV1) is cruising on the road with a little lower speed than ego vehicle(EV)(lower relative speed)
- Ego Vehicle enters follow mode and decelerates to match TV1 speed
- After a while, TV1 performs sudden deceleration. Current TTC is less than the threshold TTC for activation of emergency feature. Arbitration accepts the maneuver request of AEB.

KPIT 7/12/2023 10

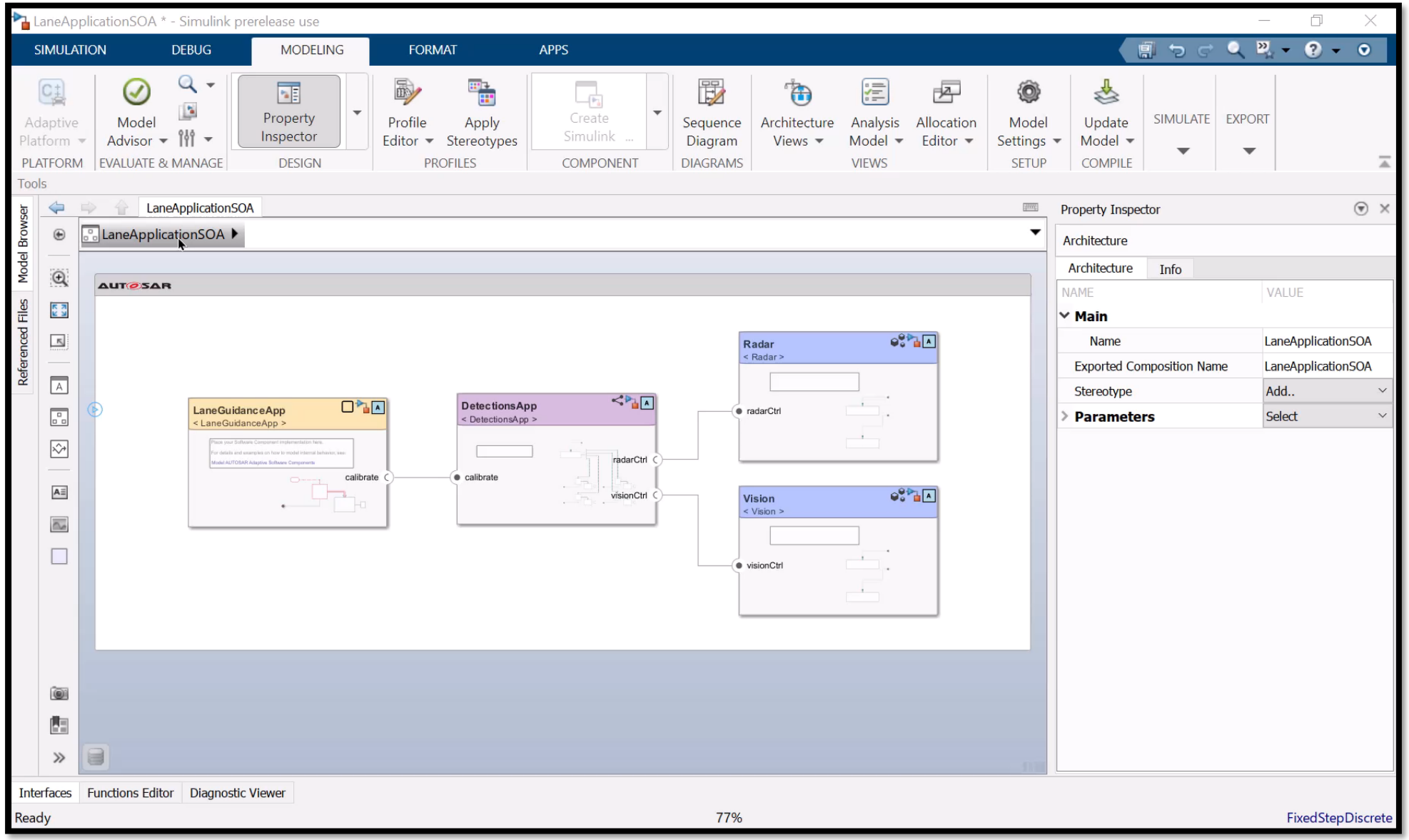
Middleware Communication Interfaces in Simulink

Categories				Simulink Construct
Communication Interfaces	Shared Memory	Send/ Write		
		Receive/ Read		
	Messaging	Send/ Publish		
		Receive/ Subscribe	Polling	
			Event-Driven	
	Remote Procedure Call	Client	Synchronous	
			Asynchronous	
		Server		

Implement and Deploy Services

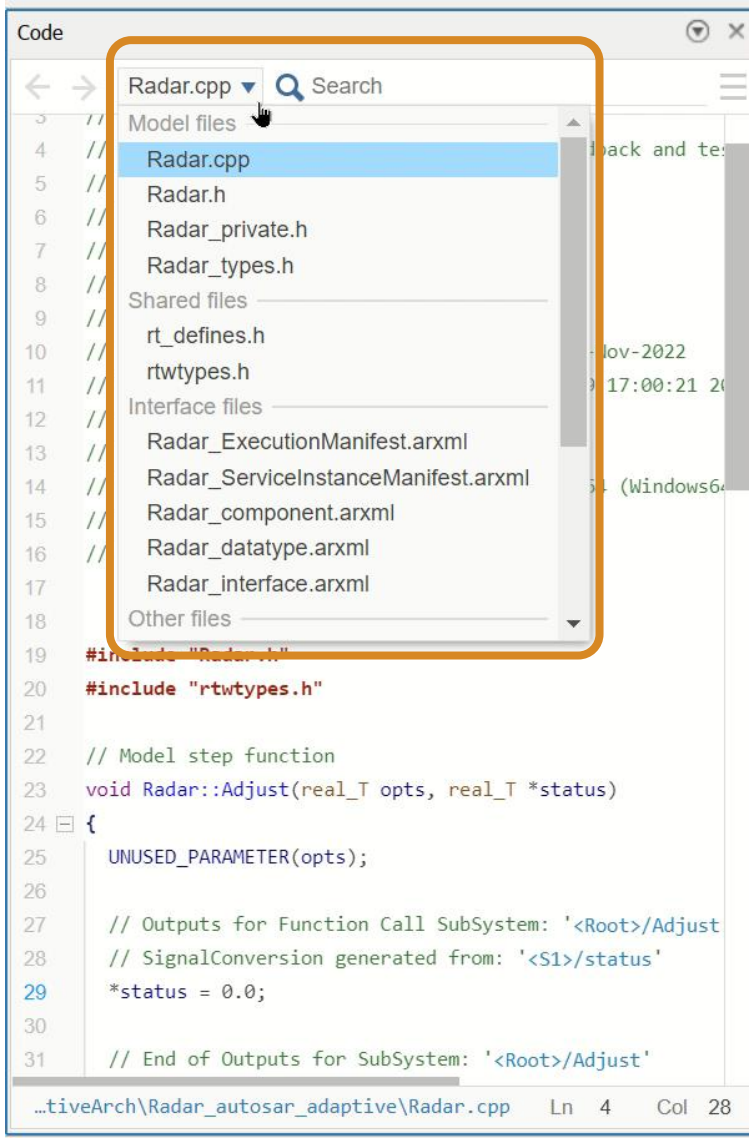
Identify and Analyze Services → Define Services and its interfaces → Define Service Behavior → Implement and deploy Services

- Services can be implemented based on AUTOSAR Adaptive platform using Model-Based-Design



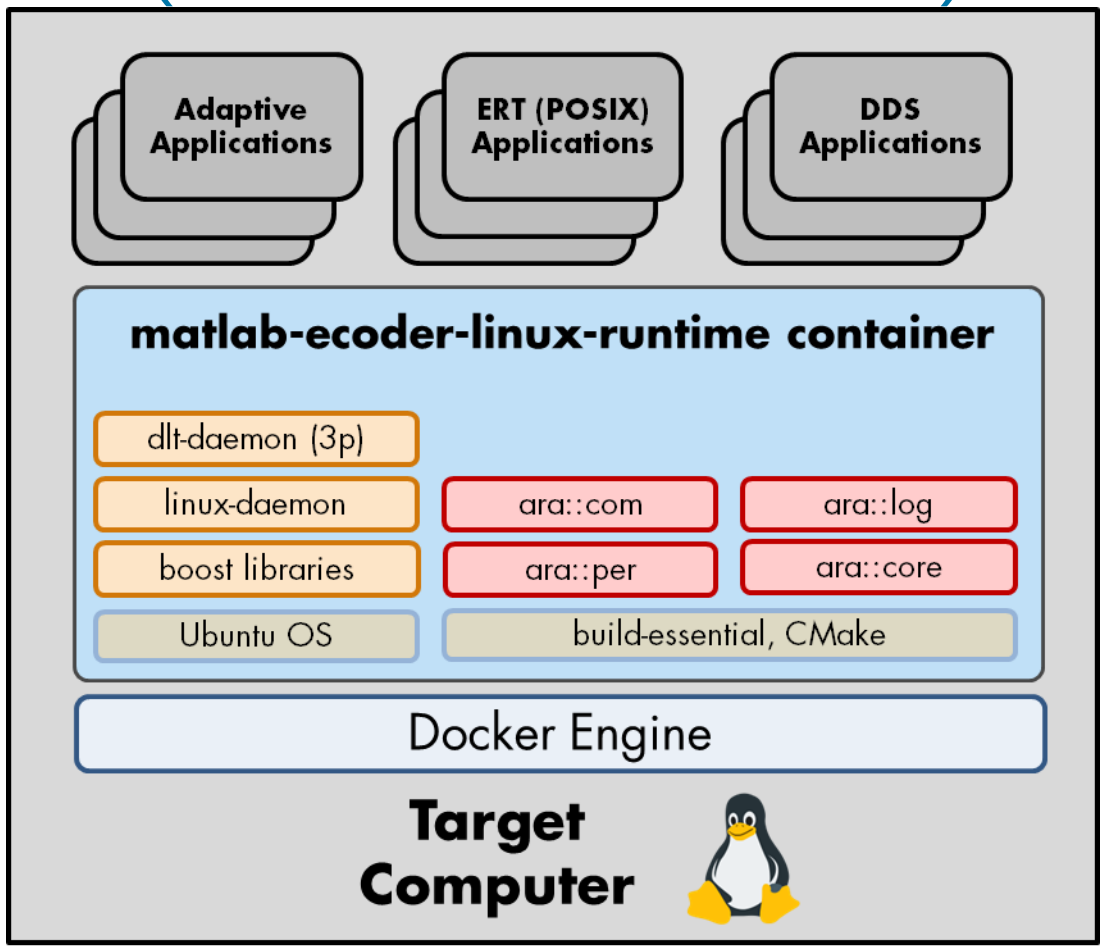
Implement and Deploy Services

- Each service need to be deployed as a standalone application, with its own artifacts including
  - Code
    - C++ Code
    - ARA Stub
  - AUTOSAR interface descriptions
    - Machine Manifest
    - Execution Manifest
    - Service Instance Manifest



Deploy, Control, and Instrument Software Applications on Linux Platform (Run-Time Environment)

- Target Linux system and Docker installation is provided by the customer
- Docker container will be installed by Support Package on target computer



Support Package for Deployment

Help Center

Search Help

CONTENTS

« Documentation Home

« Code Generation

« Embedded Coder

« Deployment, Integration, and Supported Hardware

Category

Callable Function Integration

Generated Code Interfacing

Code Packaging

Model Protection

Accelerated Simulation

Embedded Coder Support Package for Linux Applications

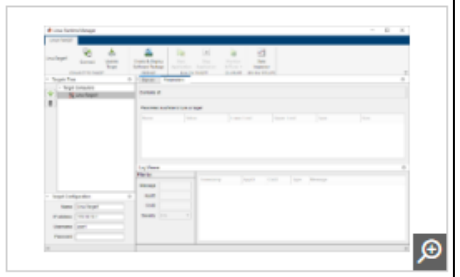
Embedded Coder Supported Hardware

Documentation | Examples | Functions | Blocks | Apps

Embedded Coder Support Package for Linux Applications

Support package to interface, manage, calibrate the Linux SOA applications

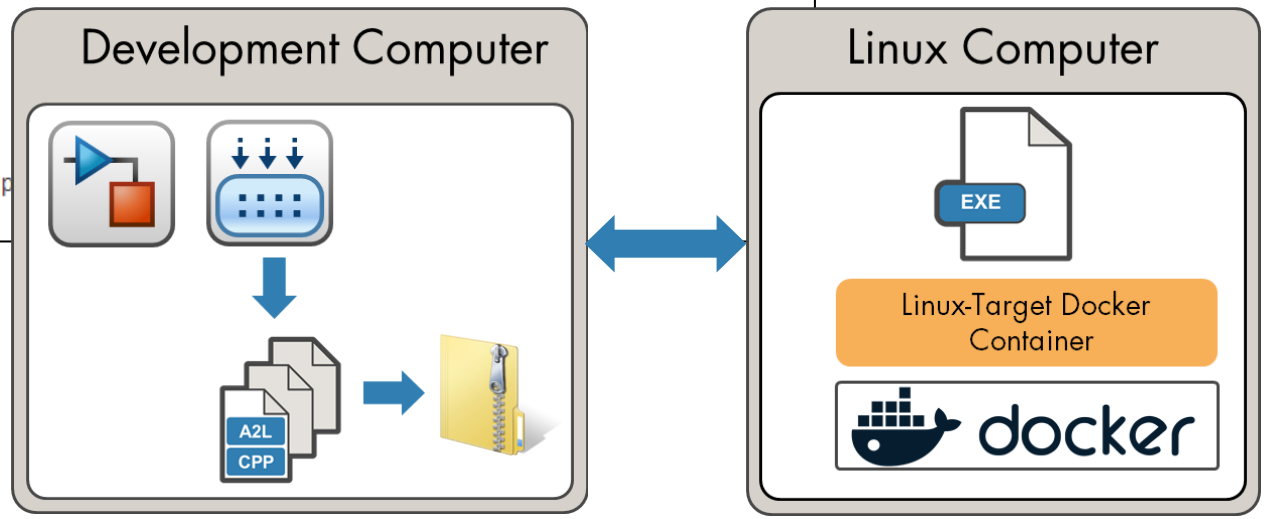
Embedded Coder® Support Package for Linux® Applications supports deploying the generated code, creating the executables, and running/stopping the executables on target, and instrument the running applications. This enables users to interact with multiple target computers at the same time.



The Embedded Coder Support Package for Linux Applications supports:

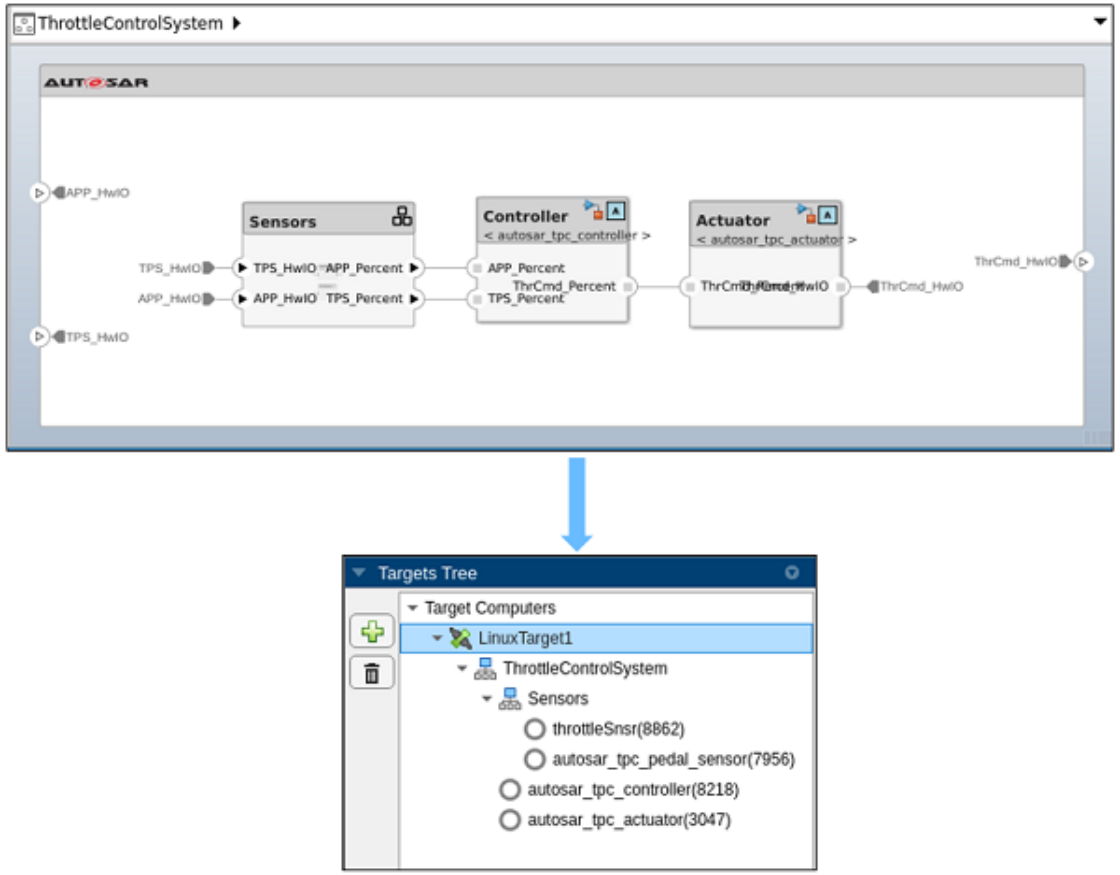
- Packaging and deployment
- Linux Runtime Manager application
- Log viewer
- Instrumentation

As part of the host services, the support package p
Runtime Manager application.



Deployment support for AUTOSAR Adaptive Architecture models

- Deploy AUTOSAR Adaptive Architecture models using Linux Support Package
- Create application package from AUTOSAR Adaptive Architecture model.
- Deploy AUTOSAR Adaptive Architecture model on Linux target.
- Start/Stop executables of AUTOSAR Adaptive Architecture application on Linux target.



Virtual ECUs deployment and testing

Challenge:

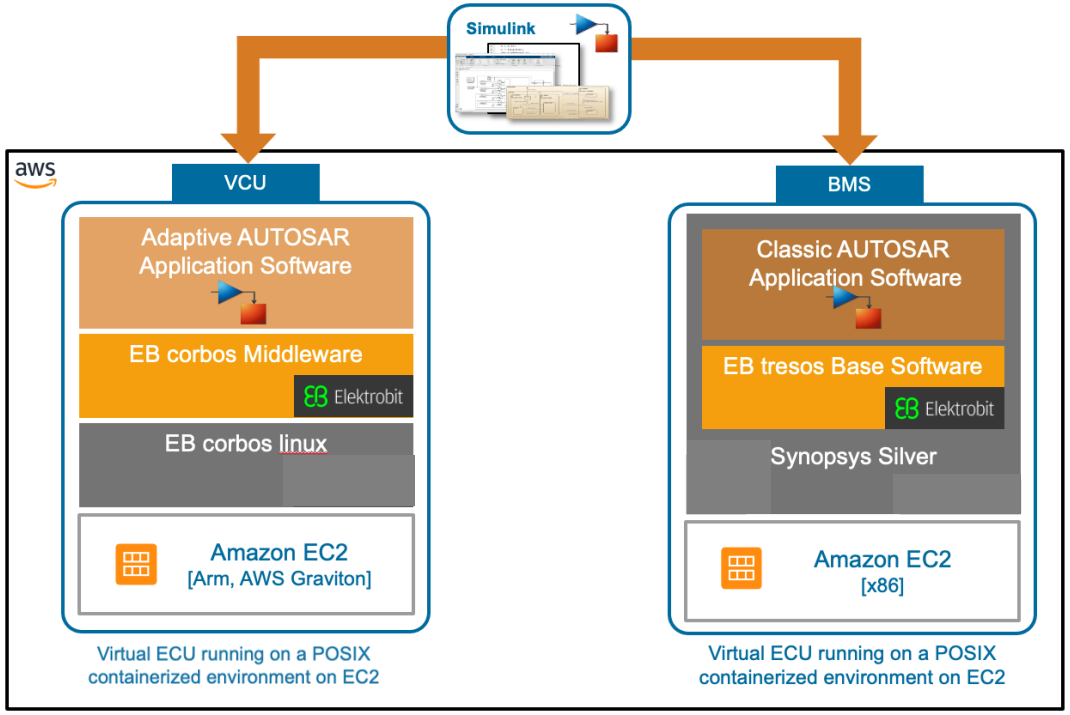
- Ensure application software can integrate into a variety of environments, depending on hardware architecture & middleware
- Verify model based AUTOSAR applications on ‘road ready’ middleware in the cloud without target hardware

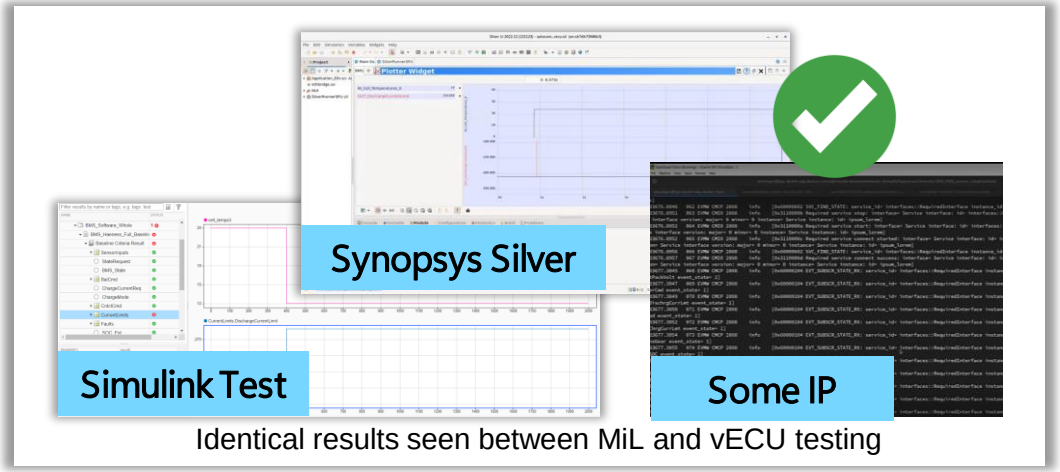
Solution:

- Customize application code interface for specific middleware/hardware
  - Classic AUTOSAR
  - Adaptive AUTOSAR / DDS / ROS

Elektrobit **Virtualize** ▪ Integrate application code with underlying middleware to deploy within virtual ECUs

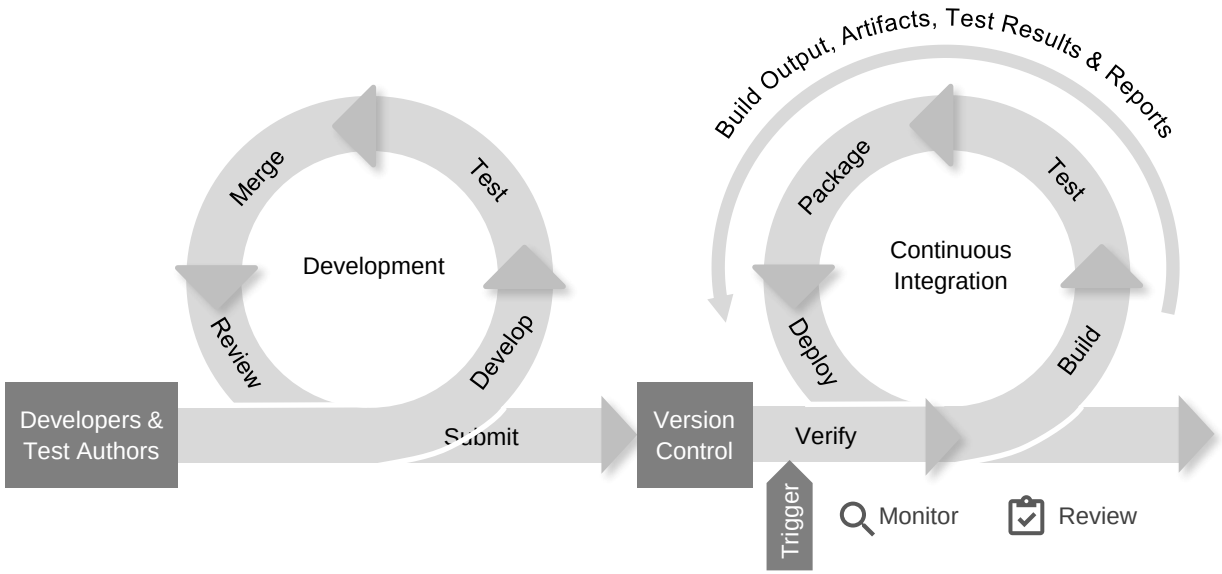
Scale ▪ Run virtual ECUs in a cloud native environment and verify the vECU is equivalent to MiL/SiL tests





Identical results seen between MiL and vECU testing

Continuous Integration



Continuous Integration (CI) originated as a software development process in which developers integrate their code into a shared repository on a regular basis.

Each commit into the shared repository is then verified by an automated build and test.

Continuous Integration Workflow with MATLAB and Simulink

Source Control Server

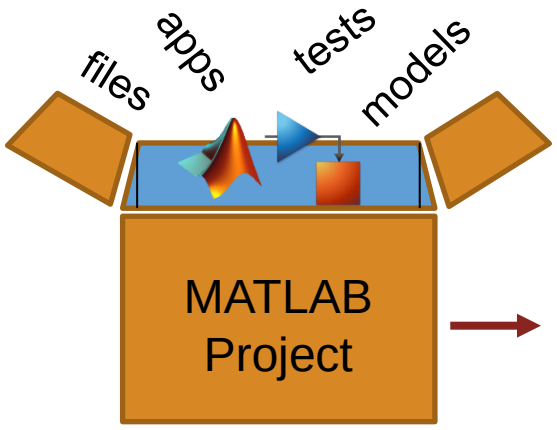
CI Server

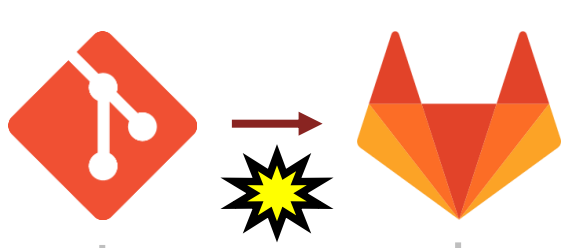
Develop | **Test** | **Build** | **Notify and Deploy**

- *Run tests:*
  - ✓ *MATLAB Unit Tests*
  - ✓ *Simulink Test*

- *Compile MEX*
- *Generate Code*
- *Package (Toolboxes, Apps)*

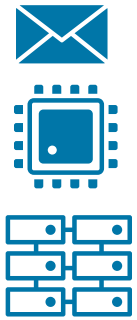
- *Publish reports*
- *Email Notification*
- *Publish to Server*
- *Hardware*





Tests pass?

Java
MATLAB
C/C++
Python
.NET
Excel
dll
COM



Commit and push changes to Git

GitLab triggers Jenkins

Jenkins run tests

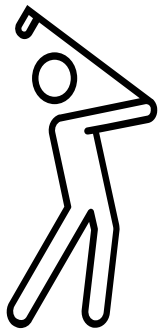
Build, generate code and package

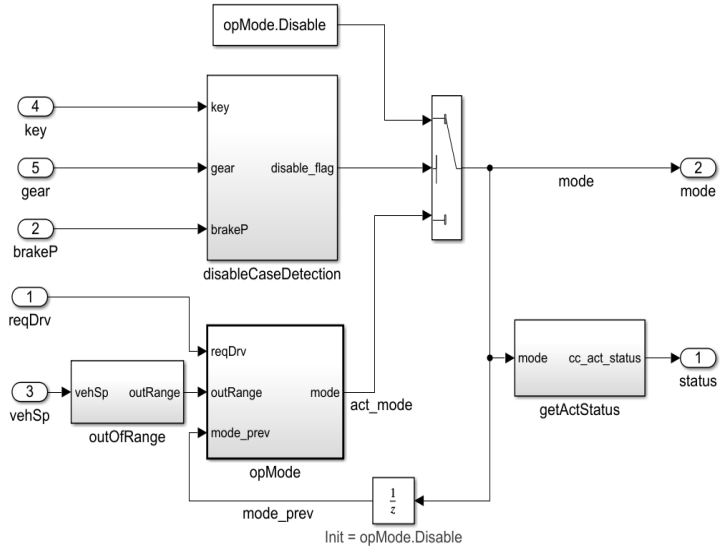
Case Study: Cruise Control System

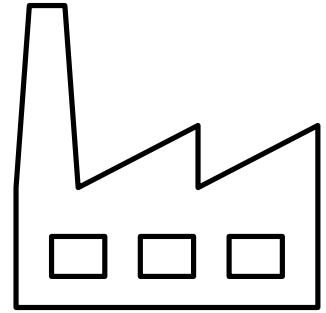
A Cruise Control Product must meet a new Driver Awareness requirement:

If the "Driver Awareness" signal is false, then:

1. Cruise control function shall be disabled.
2. And Enable shall be prevented.







Is the driver aware?

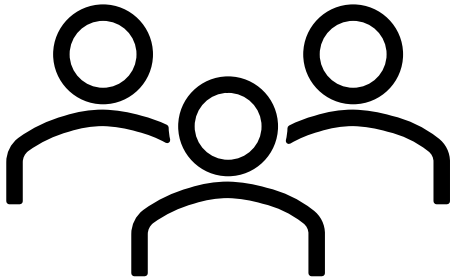
Modify logic and tests

Production

Learn More: The MBD Artifacts and YAML files for this case study can be found here

Case Study: Plan

- A team is formed

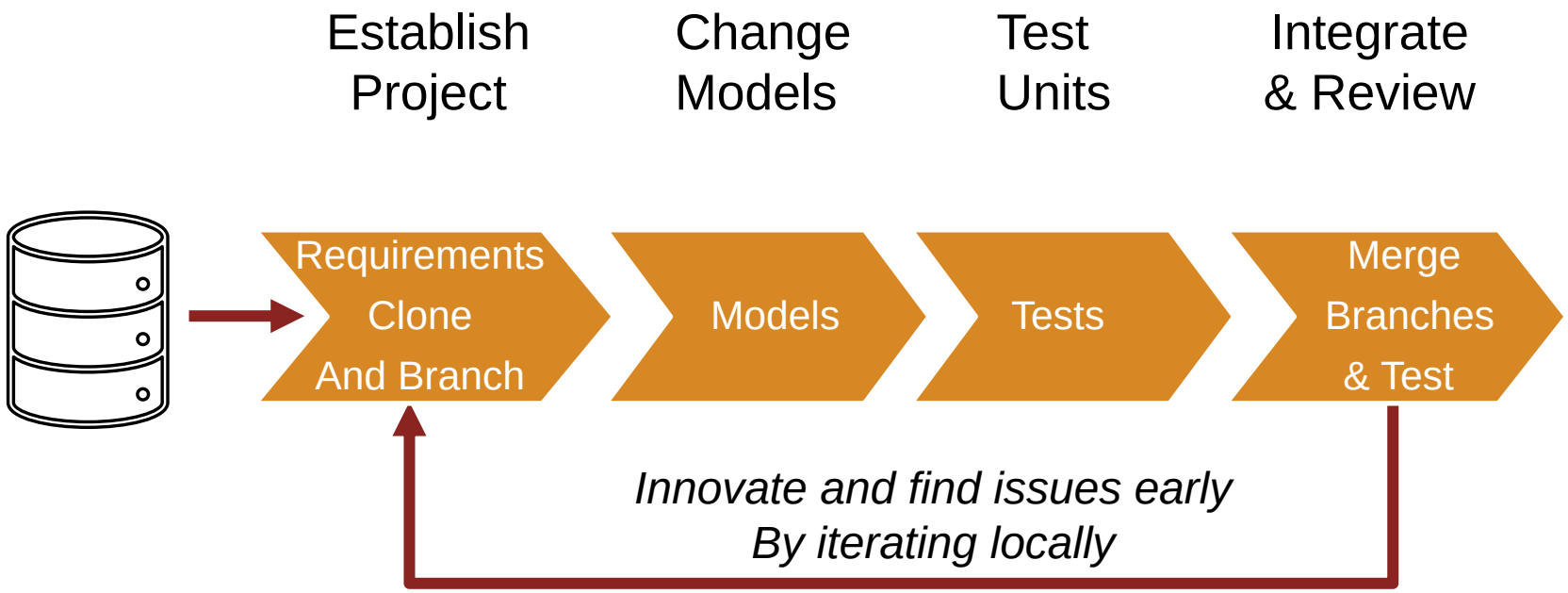


- **Team leader**, Source Control, & Integration
- **Controls Engineer 1** for Prevent Enable
- **Controls Engineer 2** for Disable
- **Test Engineer**

- The Driver Awareness feature must be released within weeks
- Fortunately, the process had been automated → speed and quality

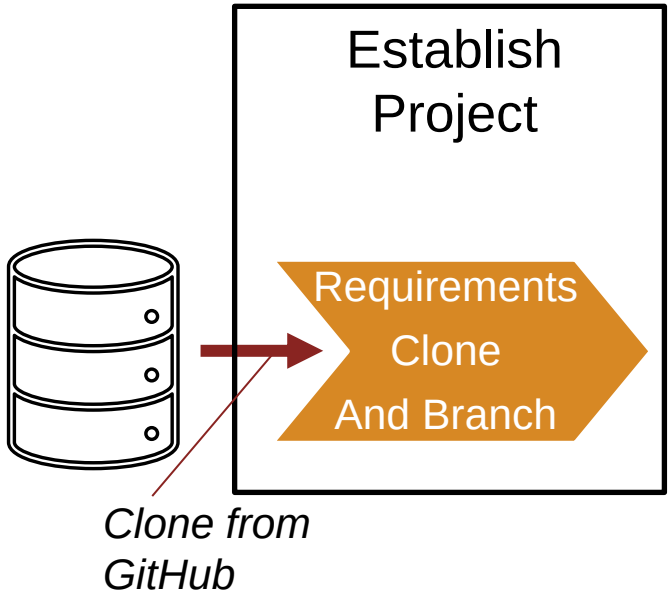
Takeaway #1: Iterate locally to innovate and prequalify

General Example



Takeaway #1: Iterate locally to innovate and prequalify

Cruise Control Case Study: Clone project and create branches



https://www.mathworks.com/help/matlab/source-control.html

Review Programmatic Operations

I will run Sections of this script

To perform most operations programmatically.

2. Link with remote SCM

This informs MATLAB where the work will be pushed to for pipeline operations

>> Copy/ paste this GitLab project URL into the "Project Remote:

https://insidelabs-git.mathworks.com/bjohnson/demo_2023_a.git

>> Validate the URL

3. Sign in to the GitLab remote if needed

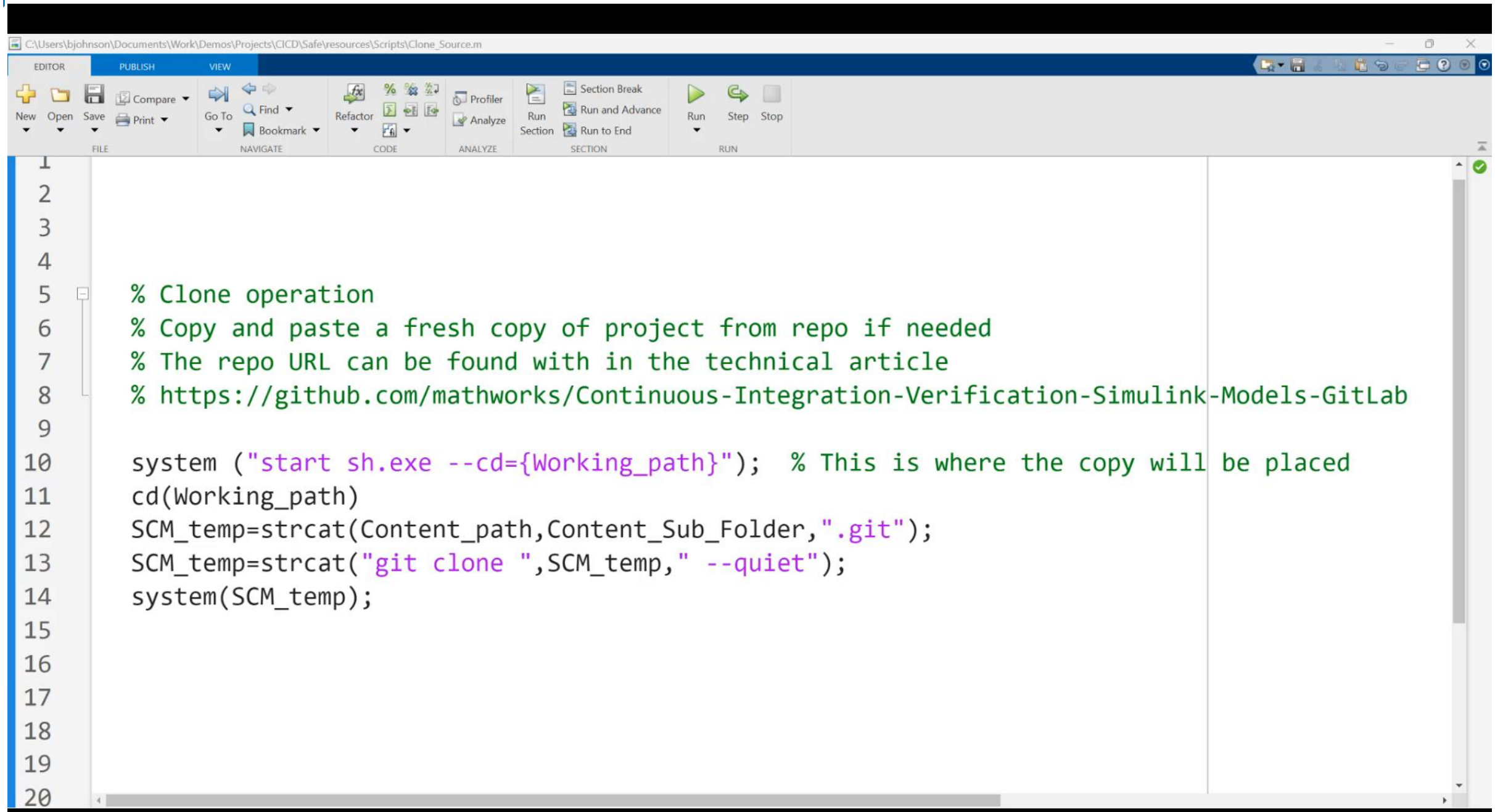
MUST - Run Section

```
9  GitLab_URL="https://insidelabs-git.mathworks.com/users/sign_in";
10 web(GitLab_URL);
```

MUST - Run Section

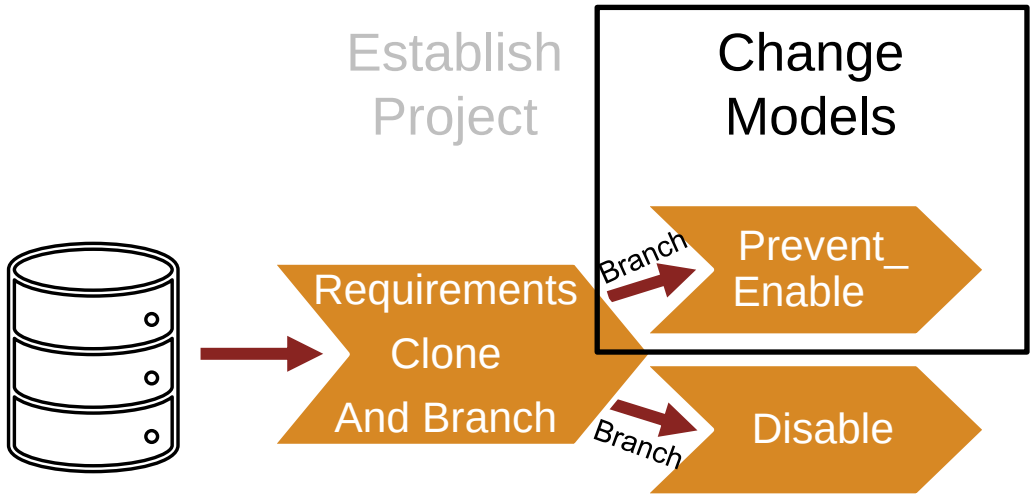
3.1 Reset the GitLab project to the fresh clone

```
11 Runner_Location='C:\Gitlab-Runner'
12 Assure_Runner_Running;
13
14 Demo=1;
15 if Demo
16 system('git add .');
17 system('git commit -am "Reset Project without running pipeline" -q');
18 system('git push -f origin main -o ci.skip'); % skip running the pipeline
19 else
20 system('git add .');
21 system('git commit -am "Reset Project and run pipeline" -q'); %q: dont echo
22 system('git push -f origin main');
23 end
```


C:\Users\bjohnson\Documents\Work\Demos\Projects\CICD\Safe\resources\Scripts\Clone_Source.m
EDITOR
PUBLISH
VIEW
New
Open
Save
Compare
Print
FILE
Go To
Find
Bookmark
NAVIGATE
Refactor
CODE
Profiler
Analyze
ANALYZE
Run Section
Section Break
Run and Advance
Run to End
SECTION
Run
Step
Stop
RUN
% Clone operation
% Copy and paste a fresh copy of project from repo if needed
% The repo URL can be found with in the technical article
% https://github.com/mathworks/Continuous-Integration-Verification-Simulink-Models-GitLab
system ("start sh.exe --cd={Working_path}"); % This is where the copy will be placed
cd(Working_path)
SCM_temp=strcat(Content_path,Content_Sub_Folder,".git");
SCM_temp=strcat("git clone ",SCM_temp," --quiet");
system(SCM_temp);

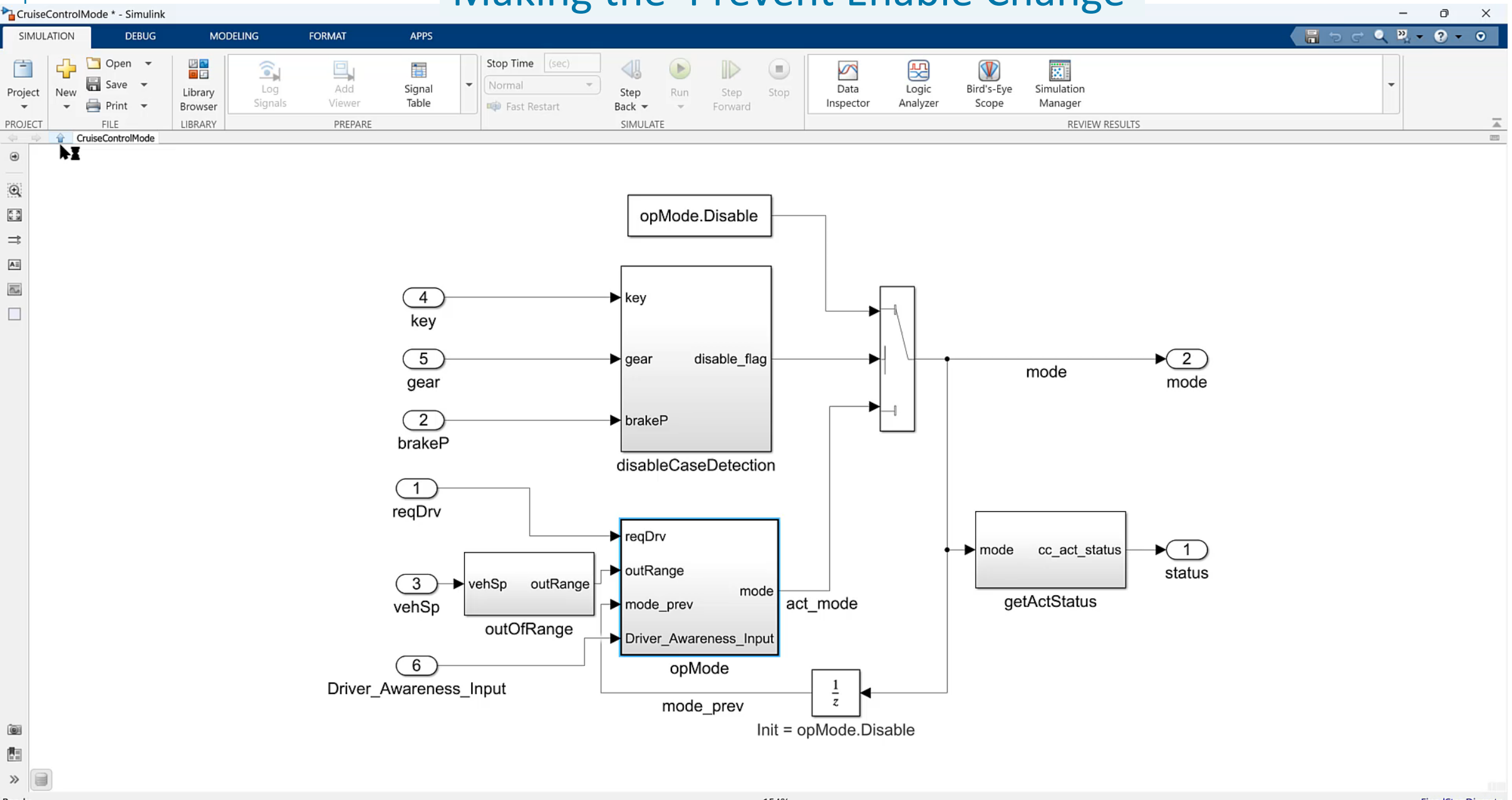
Takeaway #1: Iterate locally to innovate and prequalify

Cruise Control Case Study: Change Models



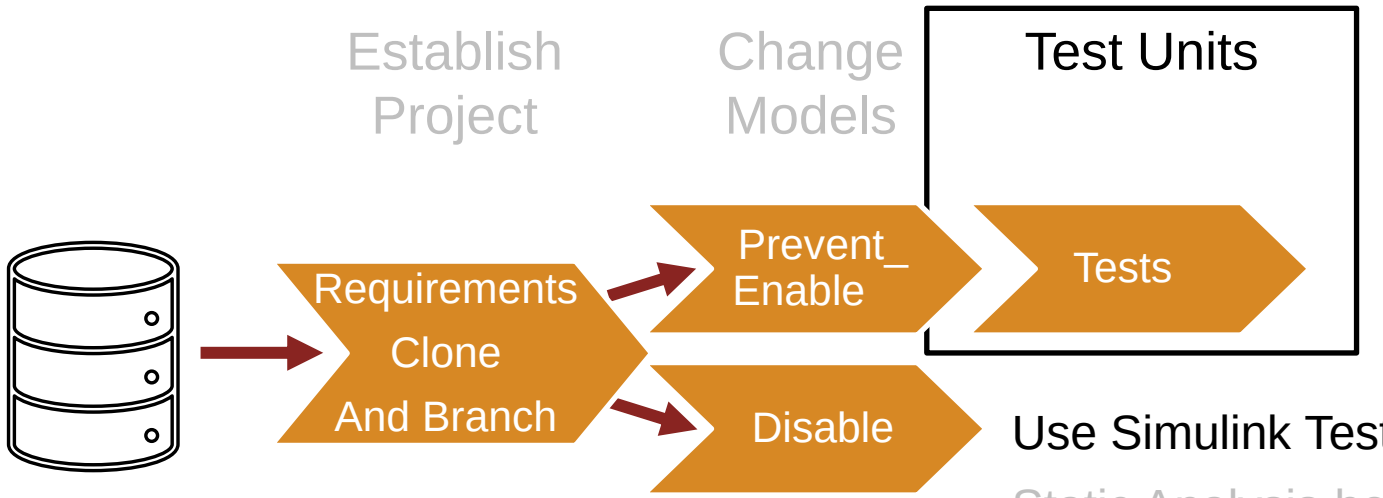
Learn More: Simulink Model Management and Architecture | Highspot

Making the Prevent Enable Change



Takeaway #1: Iterate locally to innovate and prequalify

Cruise Control Case Study: Test Units



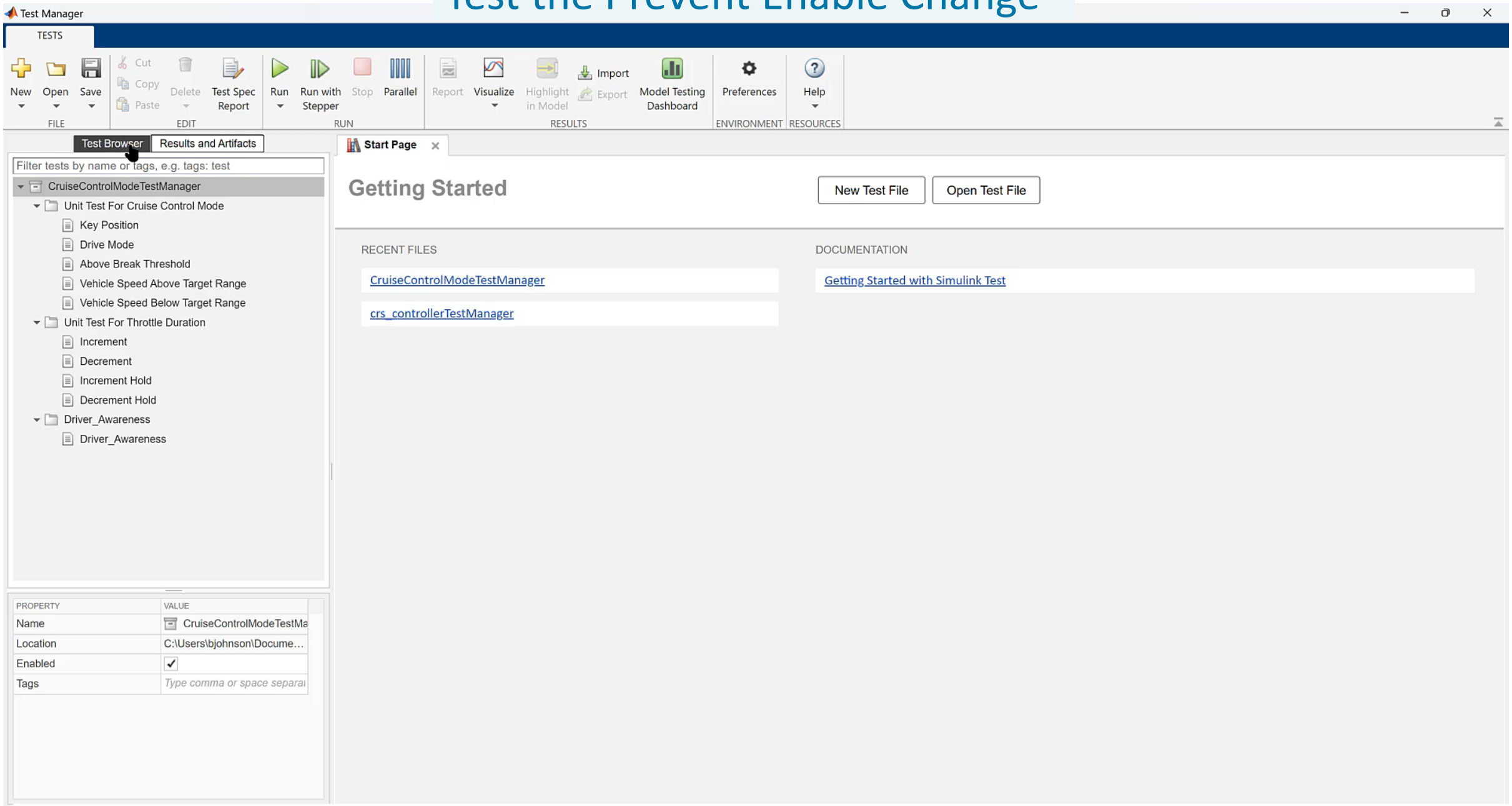
Use Simulink Test to help assure function

Static Analysis help assure structural integrity

Commit the work

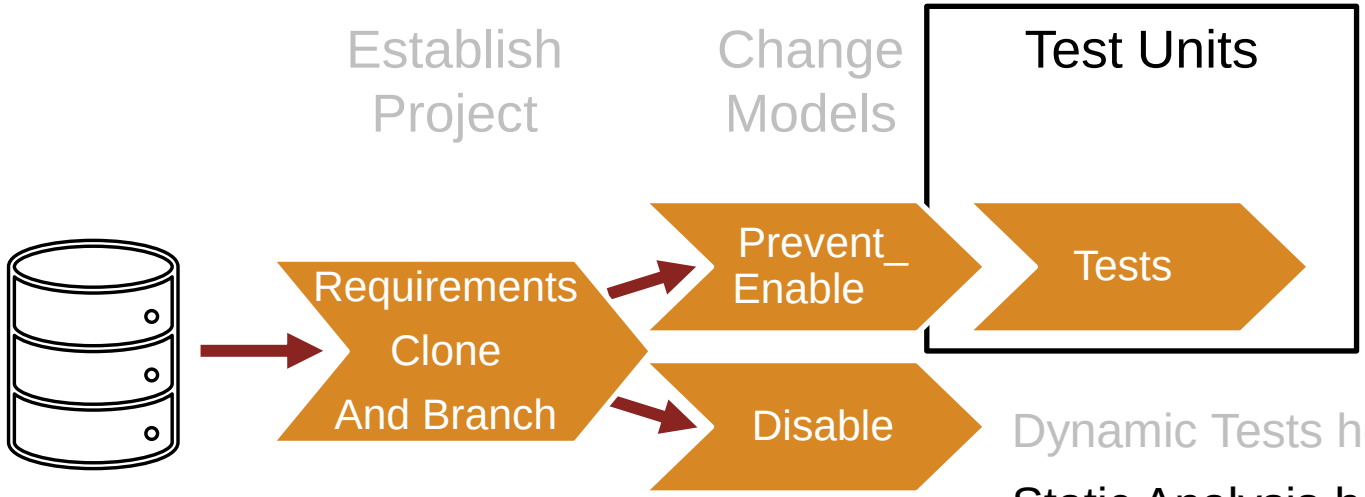
Resources: Simulink Test (mathworks.com)

Test the Prevent Enable Change



Takeaway #1: Iterate locally to innovate and prequalify

Cruise Control Case Study: Static Analysis



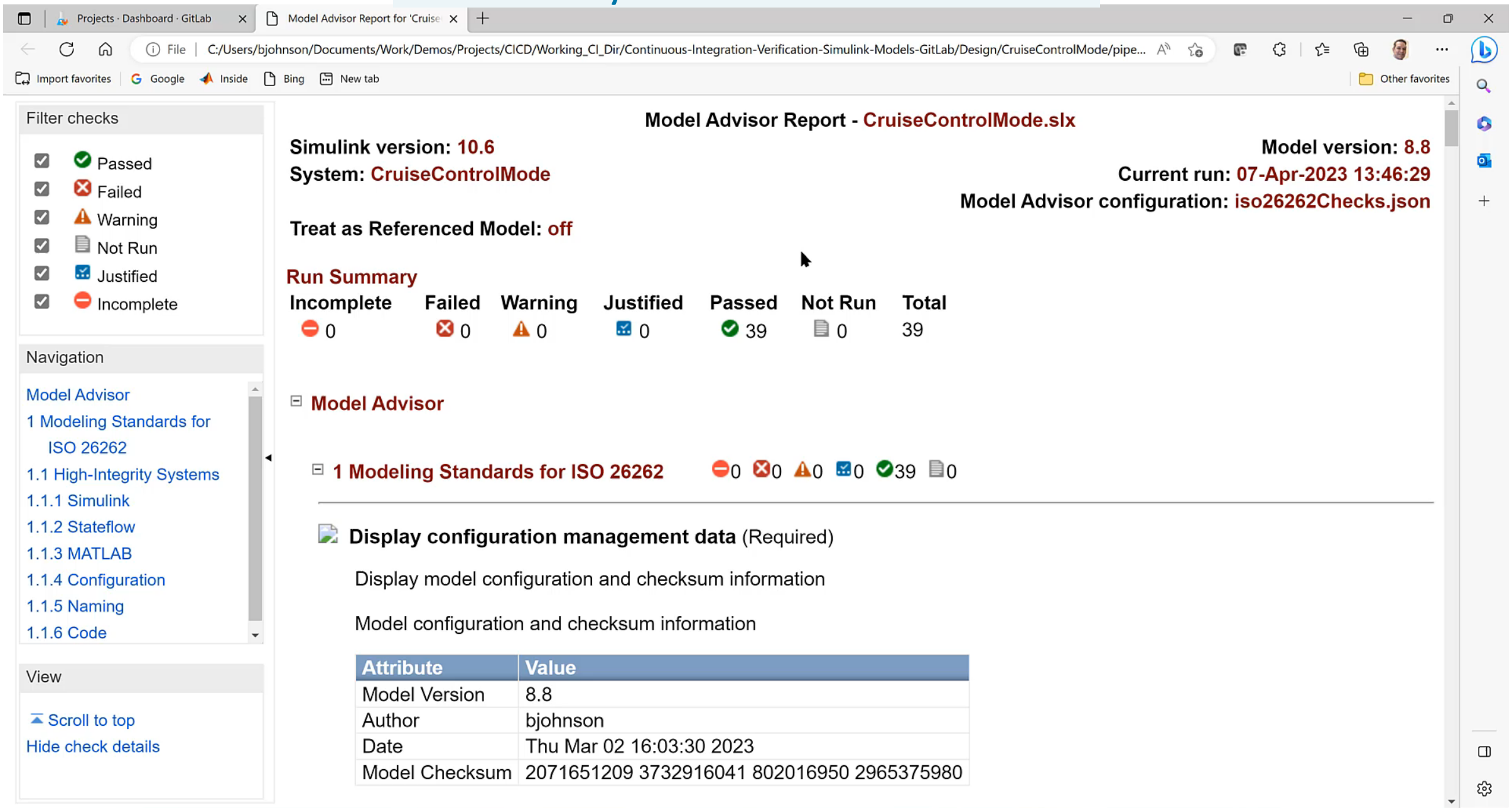
Dynamic Tests help assure function

Static Analysis help assure structural integrity

Commit the work

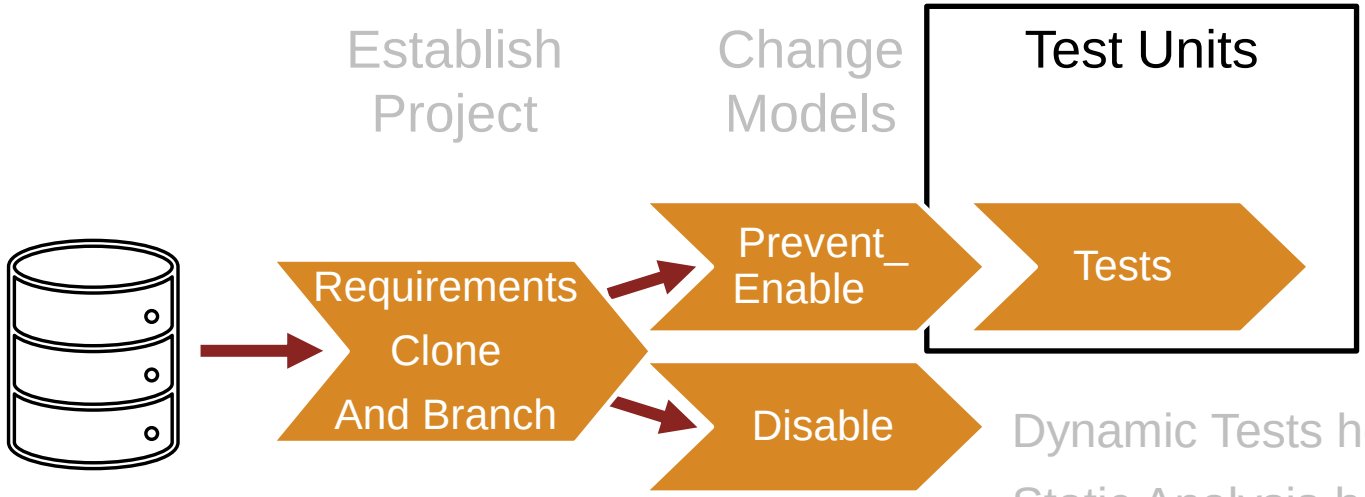
Model Advisor - MATLAB & Simulink (mathworks.com)

Static Analysis of Prevent Enable



Takeaway #1: Iterate locally to innovate and prequalify

Cruise Control Case Study: Static Analysis



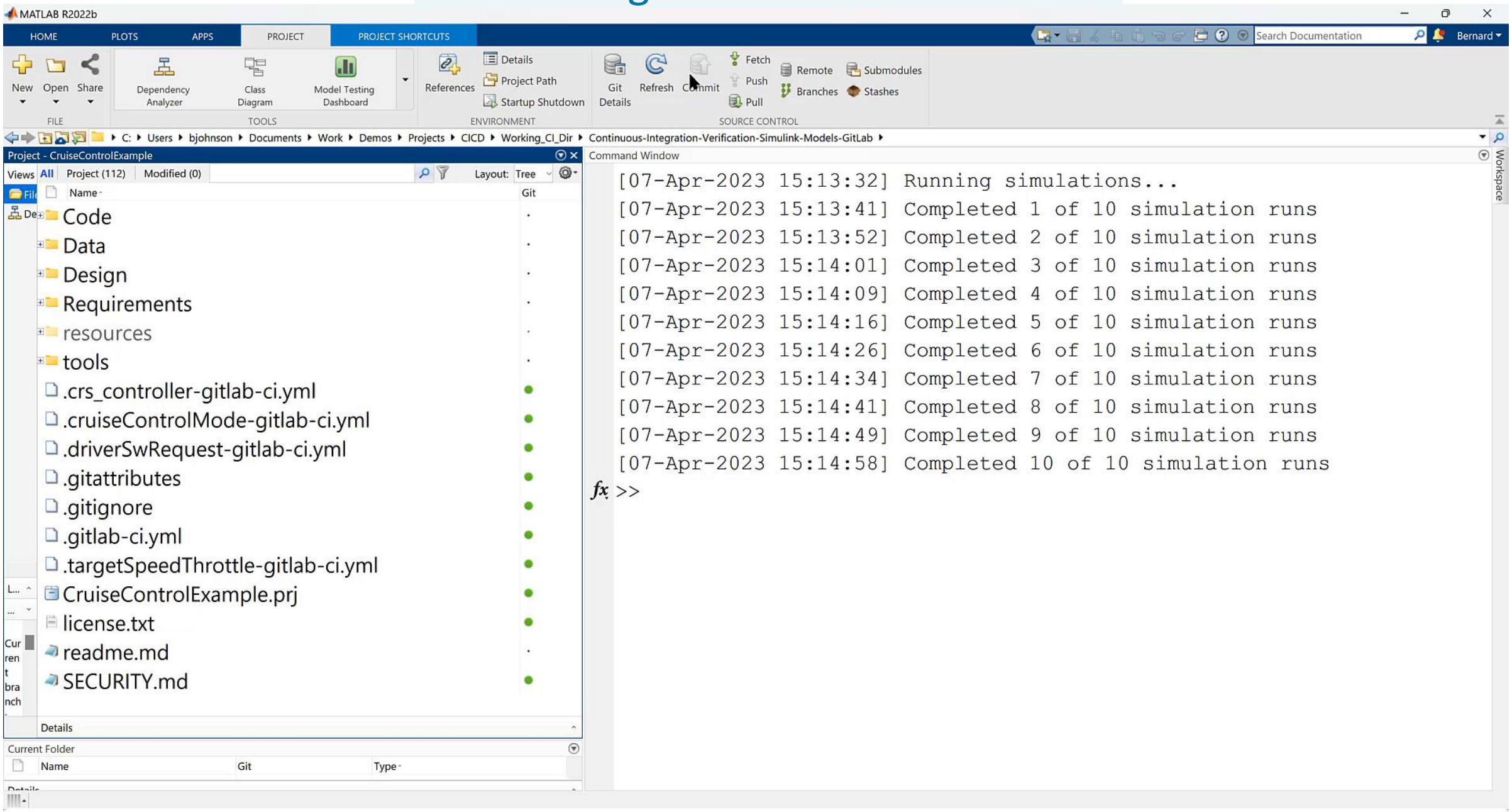
Dynamic Tests help assure function

Static Analysis help assure structural integrity

Commit the work

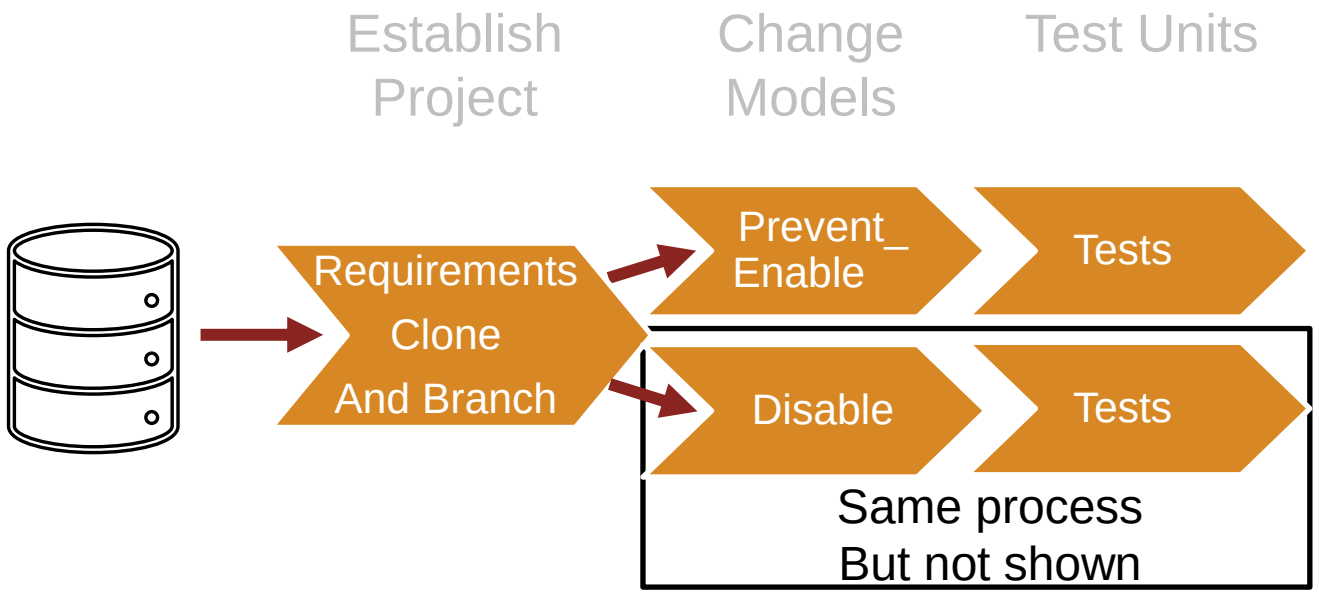
Model Advisor - MATLAB & Simulink (mathworks.com)

Committing Prevent Enable



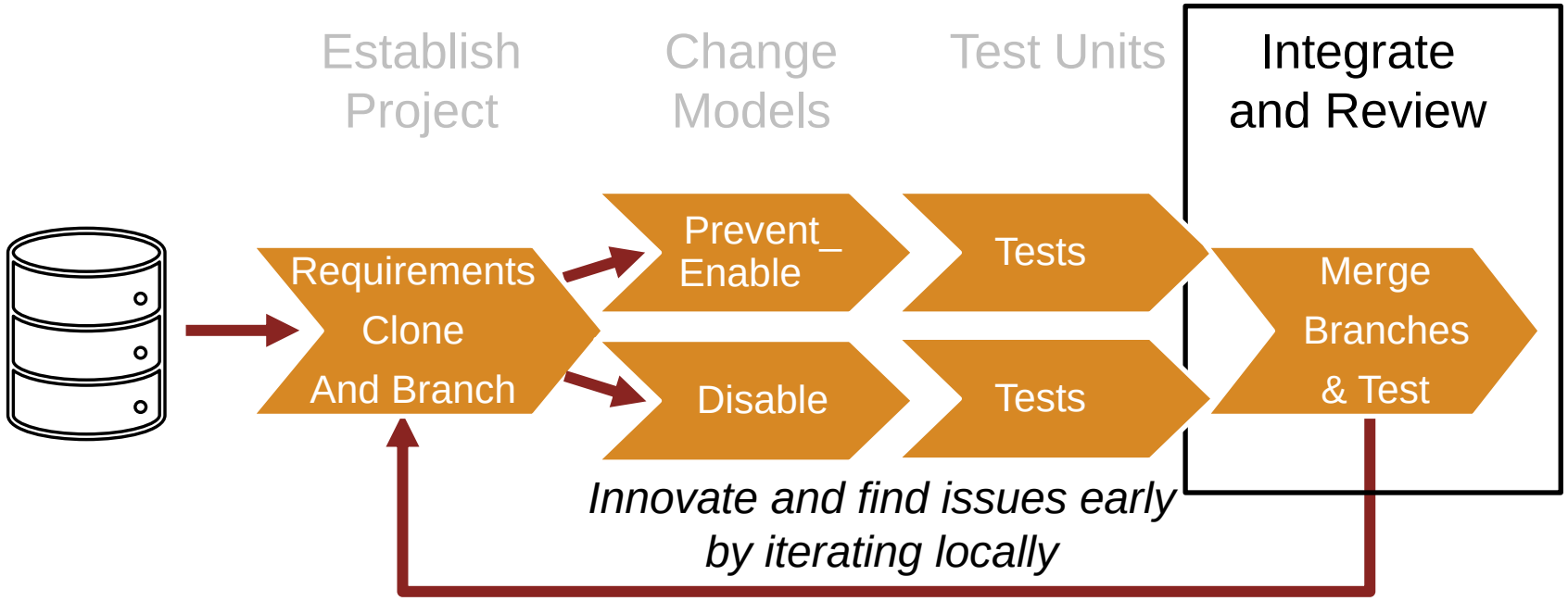
Takeaway #1: Iterate locally to innovate and prequalify

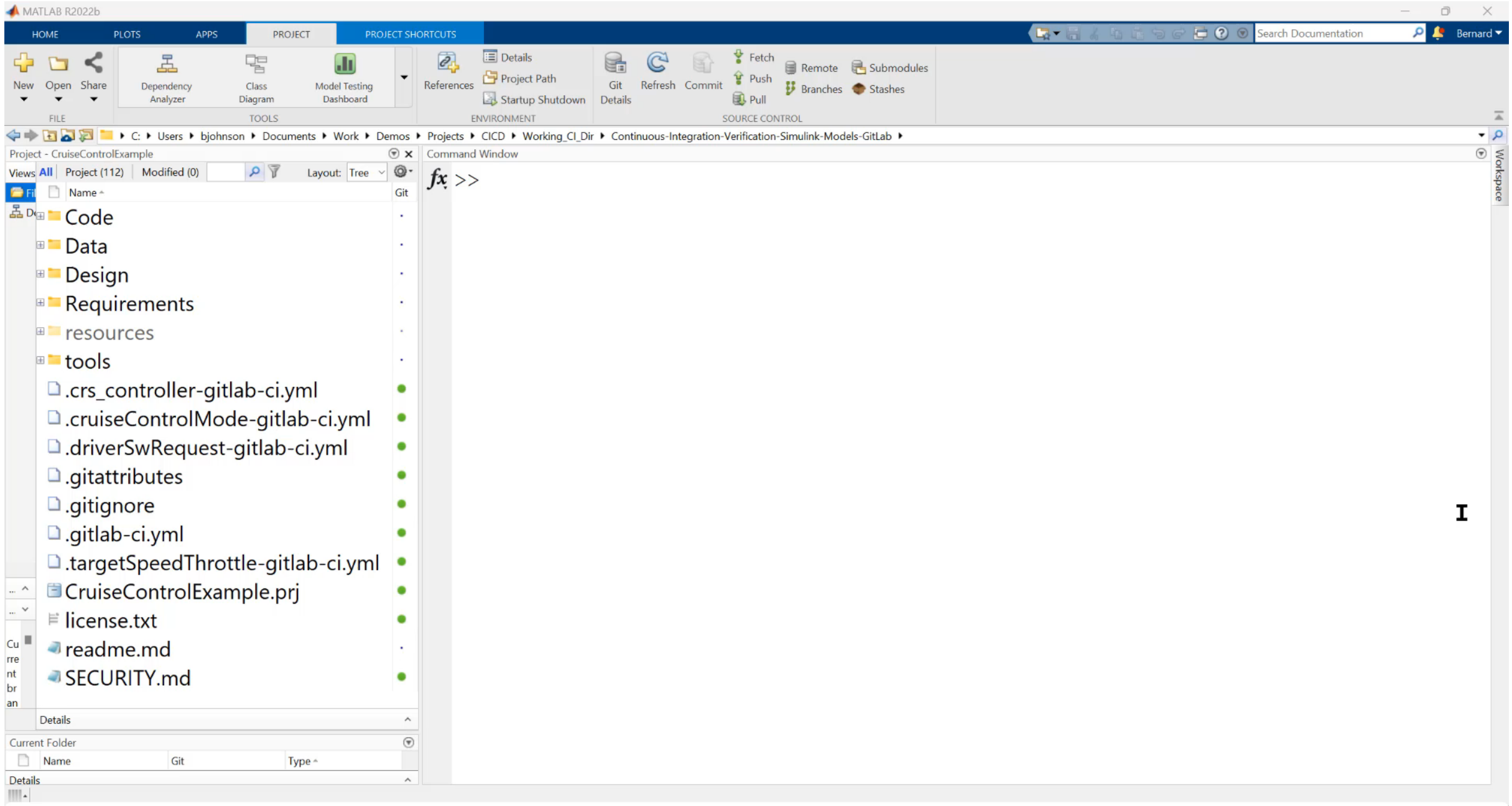
Cruise Control Case Study: Complete the Disable branch



Takeaway #1: Iterate locally to innovate and prequalify

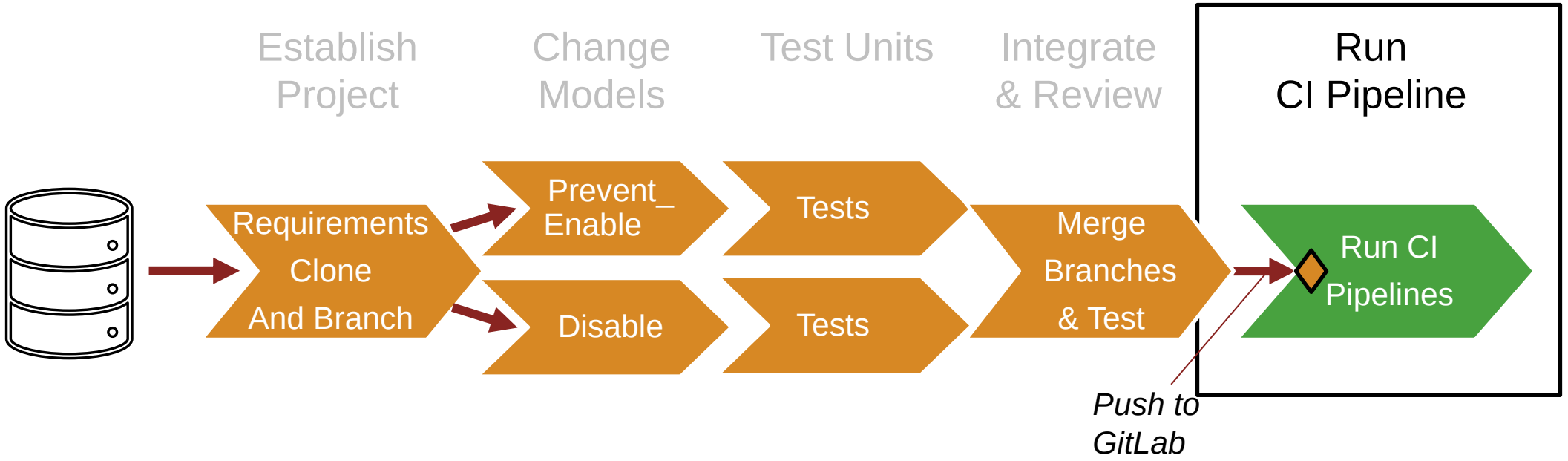
Cruise Control Case Study: Integrate and Review



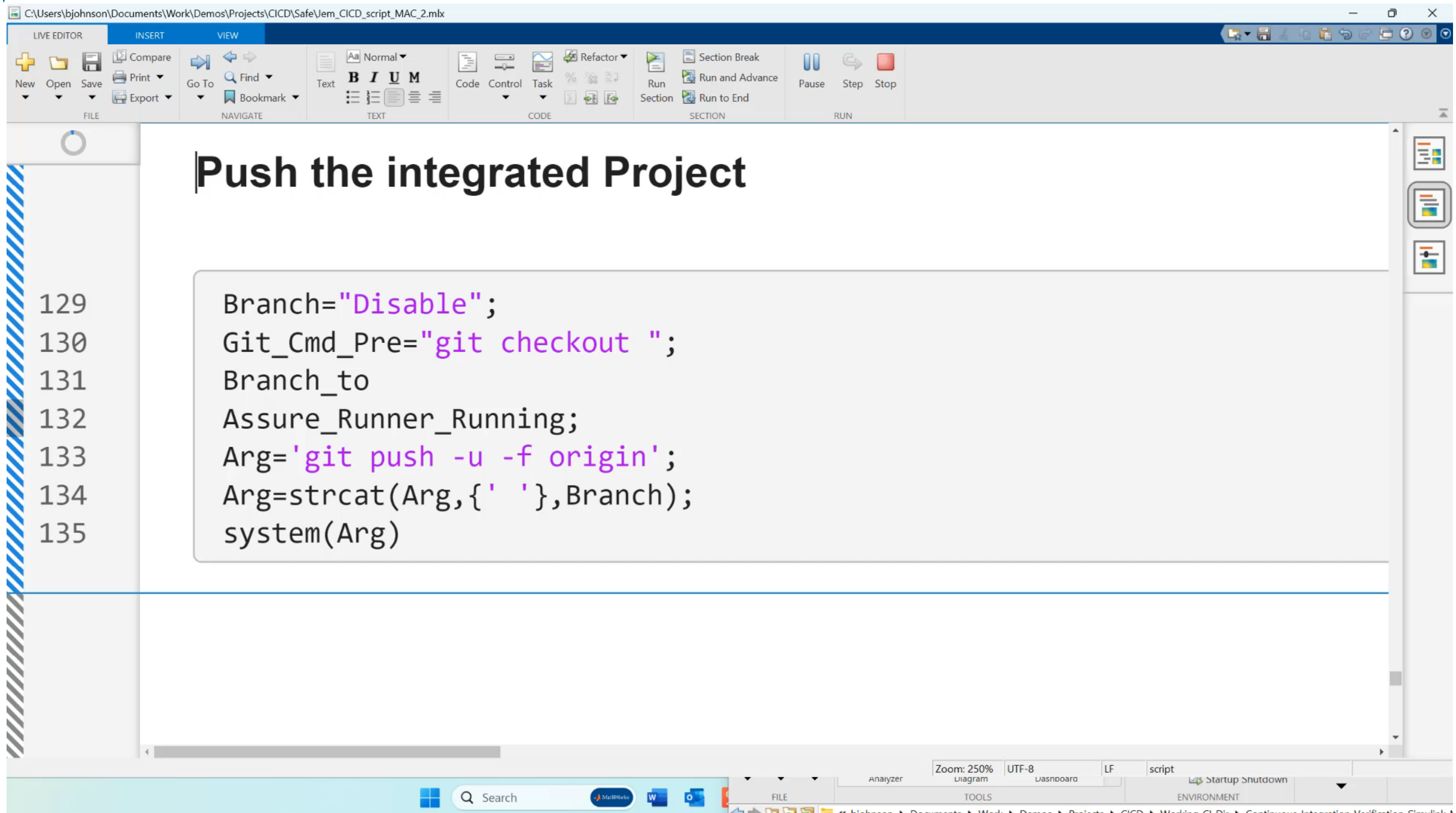

MATLAB R2022b
HOME
PLOTS
APPS
PROJECT
PROJECT SHORTCUTS
Search Documentation
Bernard
New
Open
Share
Dependency Analyzer
Class Diagram
Model Testing Dashboard
References
Details
Project Path
Startup Shutdown
Git Details
Refresh
Commit
Fetch
Push
Pull
Remote
Branches
Submodules
Stashes
FILE
TOOLS
ENVIRONMENT
SOURCE CONTROL
C: ▸ Users ▸ bjohnson ▸ Documents ▸ Work ▸ Demos ▸ Projects ▸ CICD ▸ Working_CI_Dir ▸ Continuous-Integration-Verification-Simulink-Models-GitLab ▸
Project - CruiseControlExample
Views
All
Project (112)
Modified (0)
Layout:
Tree
Name
Git
Code
Data
Design
Requirements
resources
tools
.crs_controller-gitlab-ci.yml
.cruiseControlMode-gitlab-ci.yml
.driverSwRequest-gitlab-ci.yml
.gitattributes
.gitignore
.gitlab-ci.yml
.targetSpeedThrottle-gitlab-ci.yml
CruiseControlExample.prj
license.txt
readme.md
SECURITY.md
Details
Current Folder
Name
Git
Type
Details
Command Window
>>
Workspace

Takeaway #2: Extend Model-Based Design Workflows into CI Platforms

Cruise Control Case Study: Run CI Pipeline

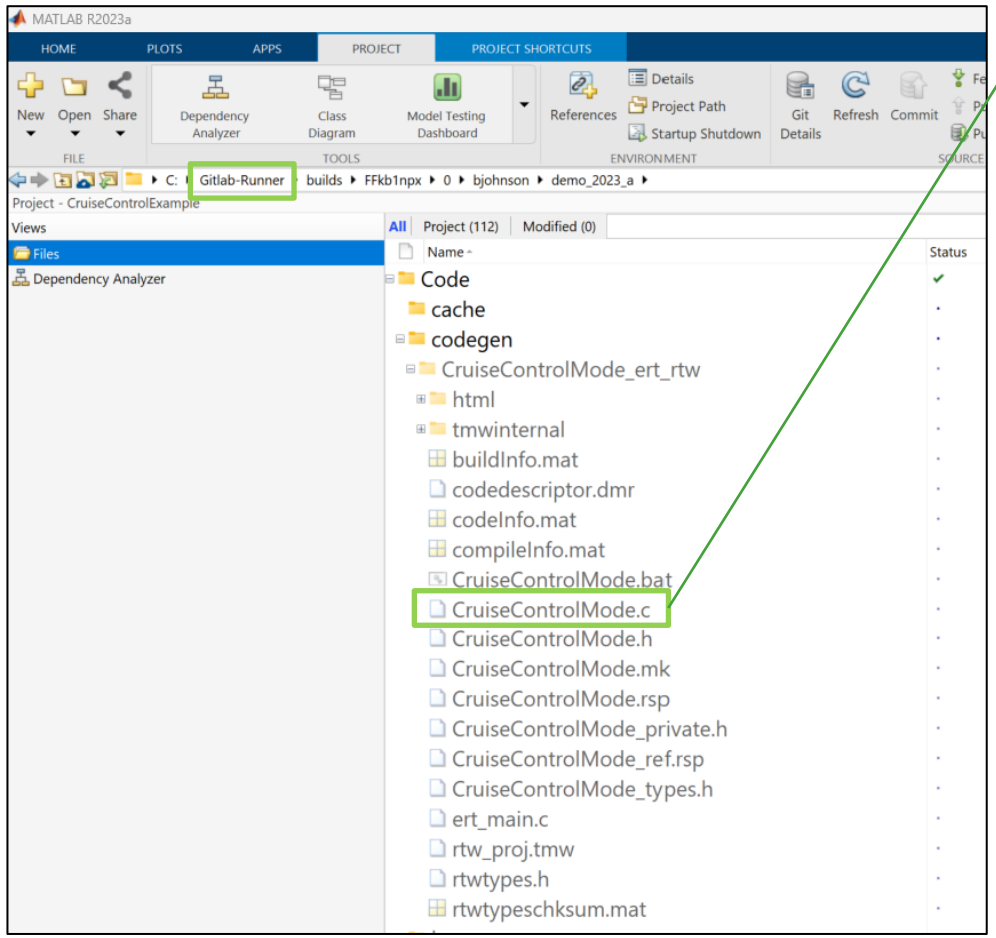


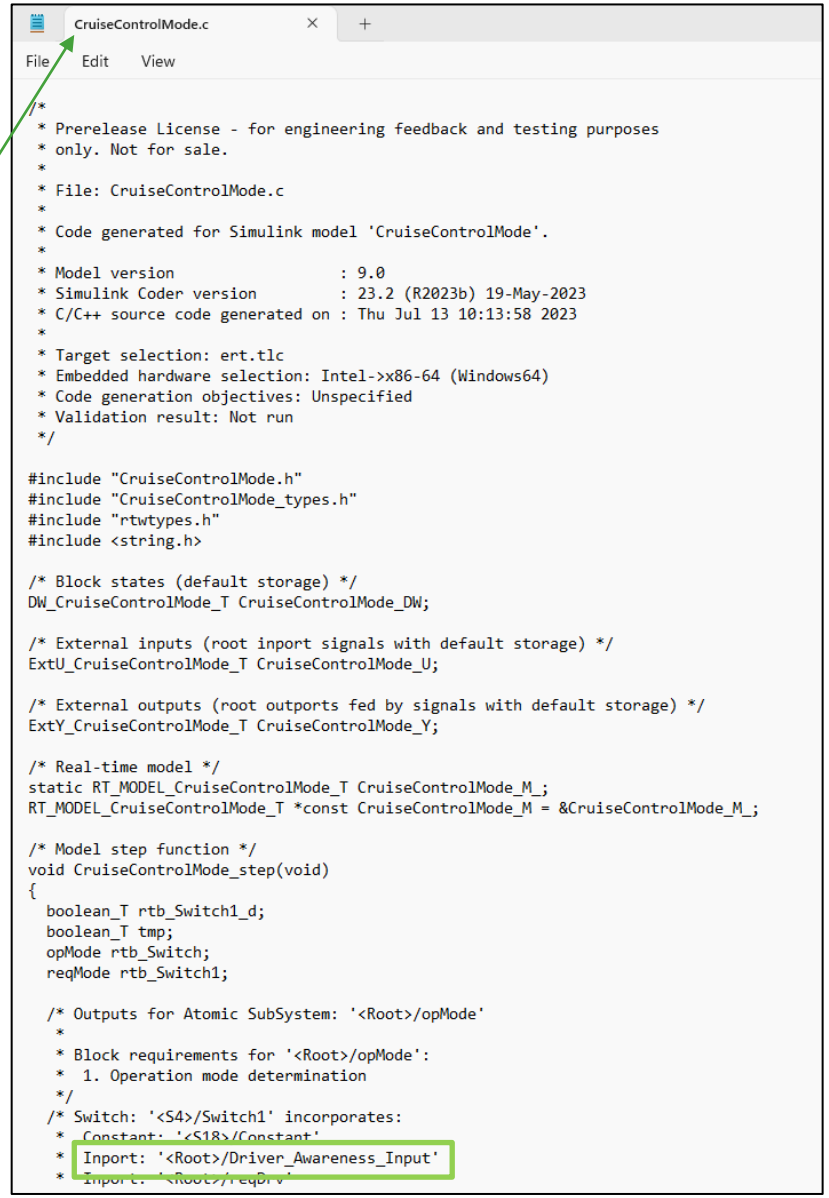
Learn More: Continuous Integration - MATLAB & Simulink (mathworks.com)


C:\Users\bjohnson\Documents\Work\Demos\Projects\CICD\Safe\Jem_CICD_script_MAC_2.mlx
LIVE EDITOR
INSERT
VIEW
New
Open
Save
Compare
Print
Export
FILE
Go To
Find
Bookmark
NAVIGATE
Text
Normal
TEXT
Code
Control
Task
Refactor
CODE
Run Section
Section Break
Run and Advance
Run to End
SECTION
Pause
Step
Stop
RUN
Push the integrated Project
129 Branch="Disable";
130 Git_Cmd_Pre="git checkout ";
131 Branch_to
132 Assure_Runner_Running;
133 Arg='git push -u -f origin';
134 Arg=strcat(Arg,{' '},Branch);
135 system(Arg)
Zoom: 250%
UTF-8
LF
script
Search
FILE
TOOLS
ENVIRONMENT

Review The Code

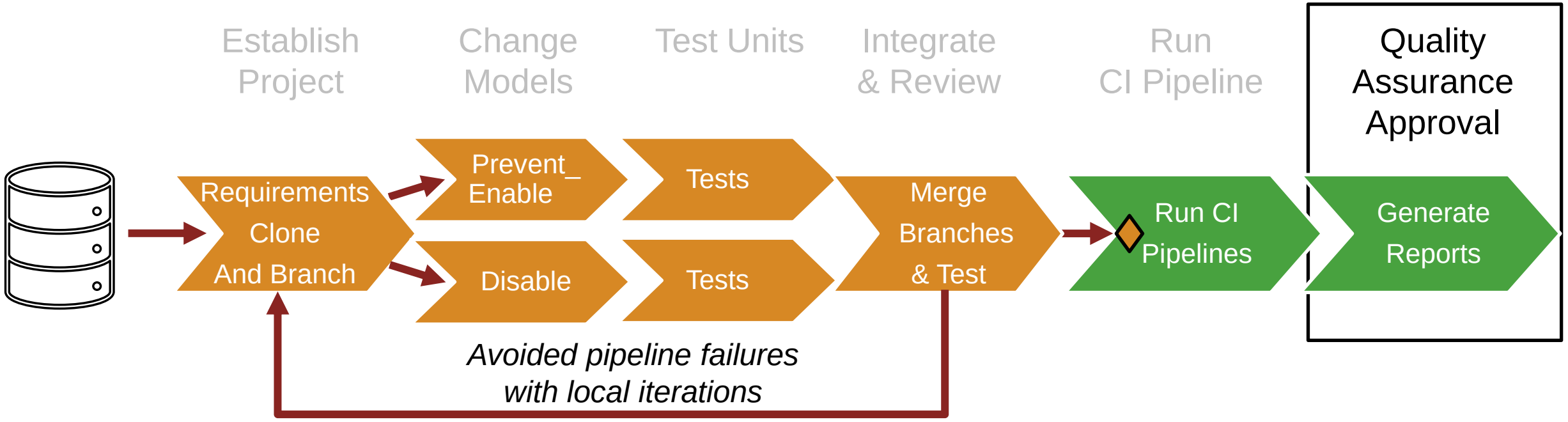
- Pipeline Outputs are deposited into the Gitlab-Runner Directory
- Consider the CruiseControlMode.c file
- Open it with an Editor
- See that the Driver Awareness change is in place

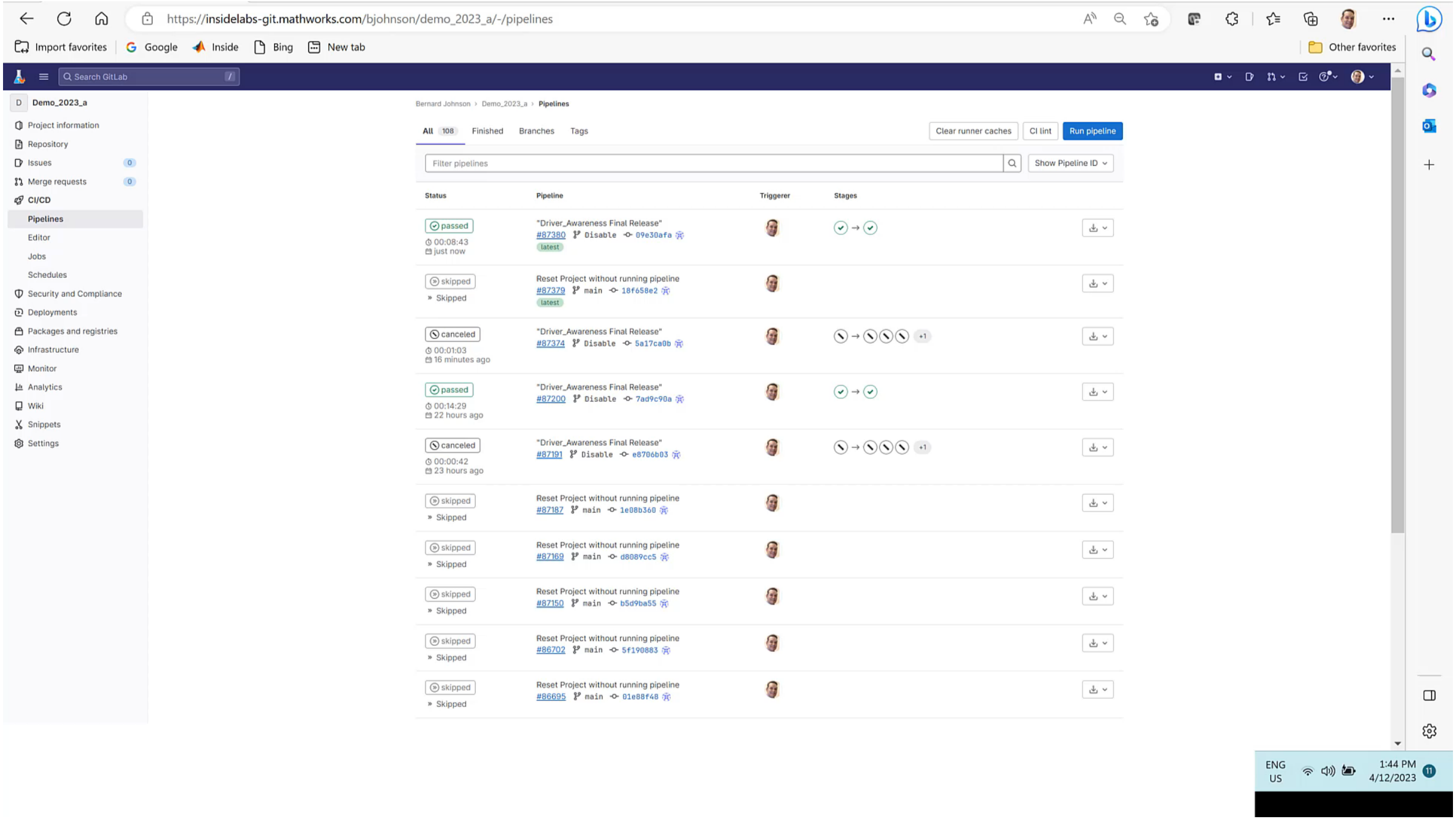




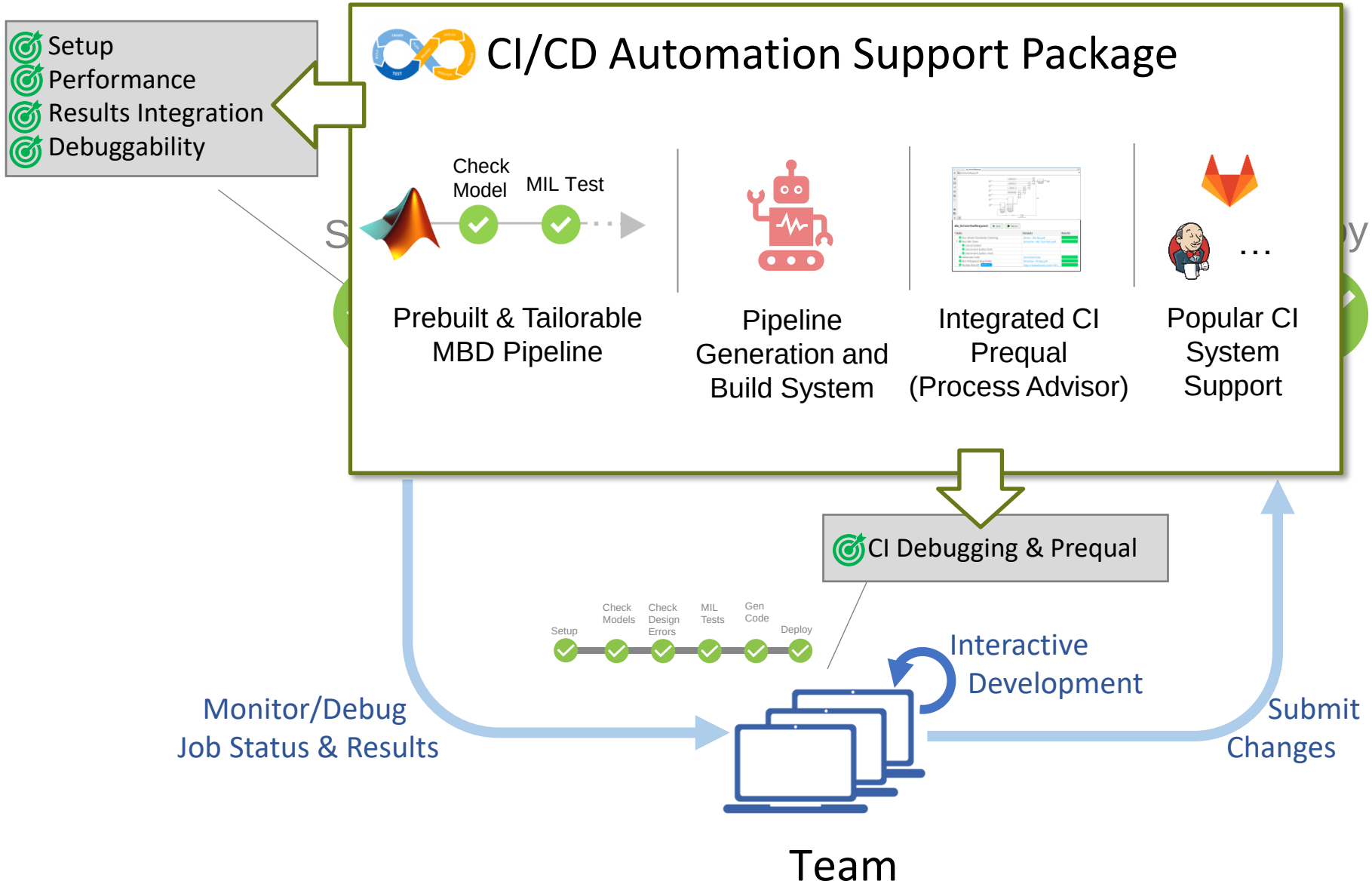
Takeaway #2: Extend Model-Based Design Workflows into CI Platforms

Cruise Control Case Study: Approve Release



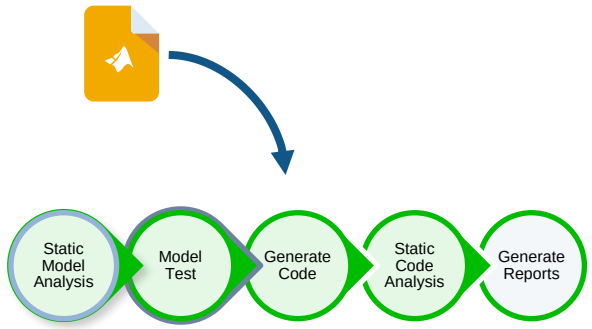

https://insidelabs-git.mathworks.com/bjohnson/demo_2023_a/-/pipelines
Import favorites
Google
Inside
Bing
New tab
Other favorites
Search GitLab
Demo_2023_a
Project information
Repository
Issues
Merge requests
CI/CD
Pipelines
Editor
Jobs
Schedules
Security and Compliance
Deployments
Packages and registries
Infrastructure
Monitor
Analytics
Wiki
Snippets
Settings
Bernard Johnson › Demo_2023_a › Pipelines
All 108
Finished
Branches
Tags
Clear runner caches
CI lint
Run pipeline
Filter pipelines
Show Pipeline ID
Status
Pipeline
Triggerer
Stages
passed
00:08:43
just now
"Driver_Awareness Final Release"
#87380 Disable 09e30afa
latest
skipped
Skipped
Reset Project without running pipeline
#87379 main 18f658e2
latest
canceled
00:01:03
16 minutes ago
"Driver_Awareness Final Release"
#87374 Disable 5a17ca0b
passed
00:14:29
22 hours ago
"Driver_Awareness Final Release"
#87200 Disable 7ad9c90a
canceled
00:00:42
23 hours ago
"Driver_Awareness Final Release"
#87191 Disable e8706b03
Reset Project without running pipeline
#87187 main 1e08b360
#87169 main d8089cc5
#87150 main b5d9ba55
#86702 main 5f190883
#86695 main 01e88f48
ENG
US
1:44 PM
4/12/2023

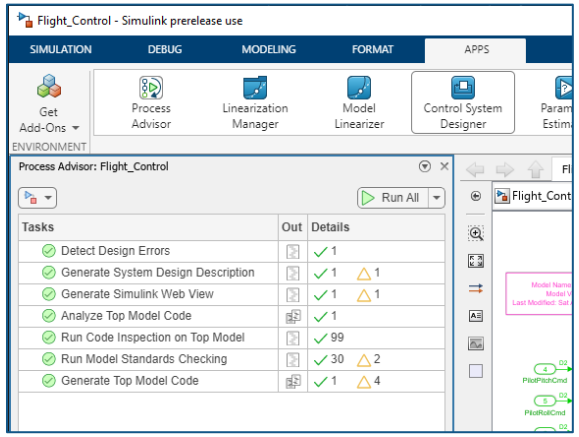
Simplifying Adoption and Optimizing CI/CD for Model-Based Design

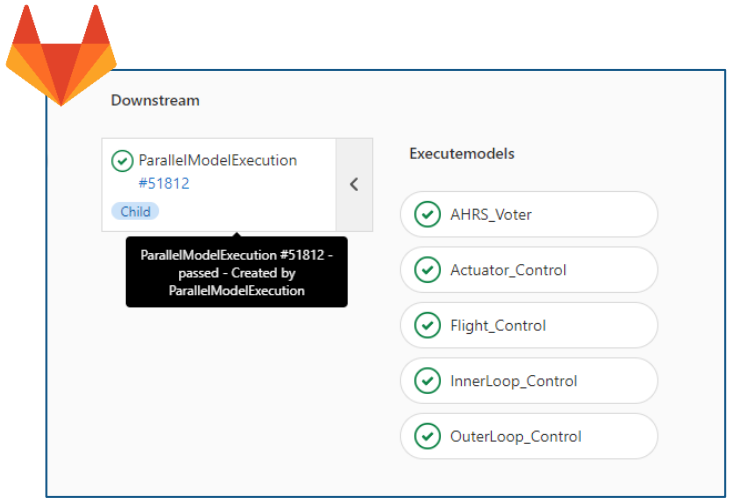


CI/CD Automation for Simulink Check Support Package

R2022a
August







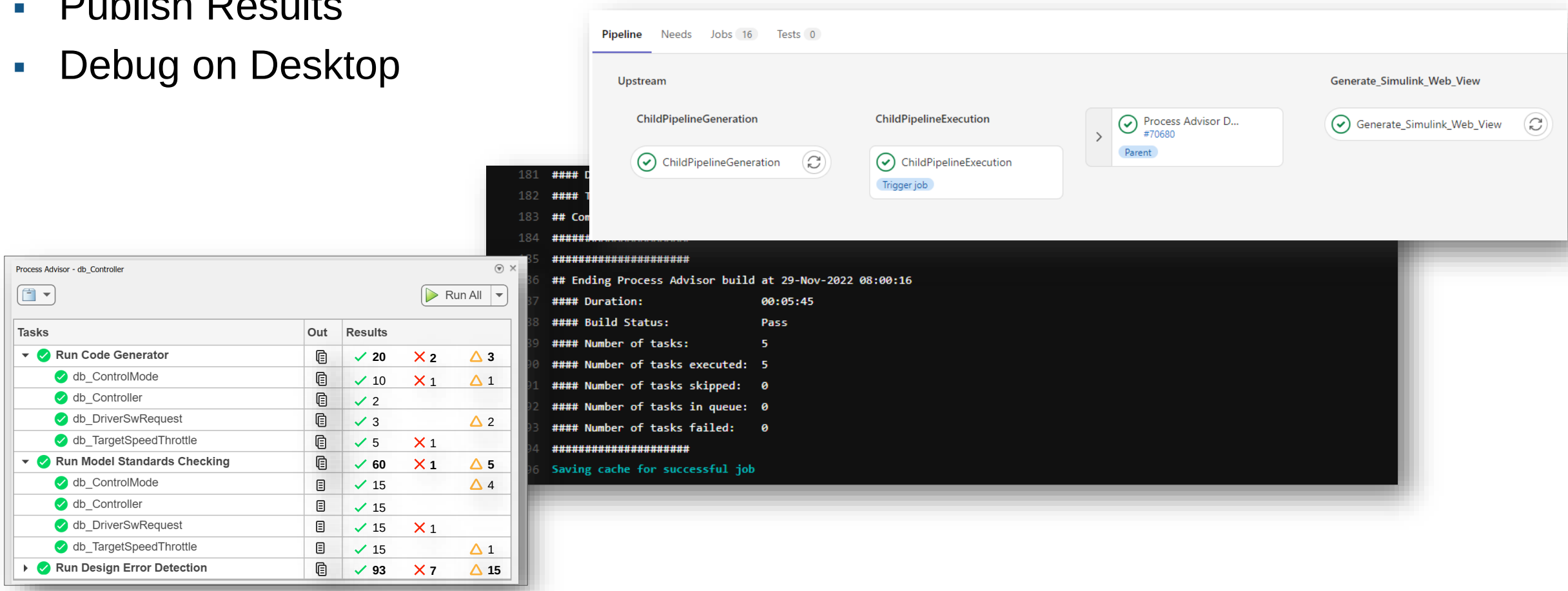
1) Simple Setup
- ✓ Prebuilt Model-Based Design pipeline
- ✓ Built-in Model-Based Design tool support
- ✓ Tailorable

2) Desktop Integration with Process Advisor app
- ✓ Local prequalification
- ✓ Local Debugging

3) 3rd Party CI Integration
- ✓ Jenkins/Gitlab YAML
- ✓ Optimized Model-Based Design Builds
- ✓ CI Results Integration

Integration with common CI Systems

- Automated Pipeline Generation
- Publish Results
- Debug on Desktop



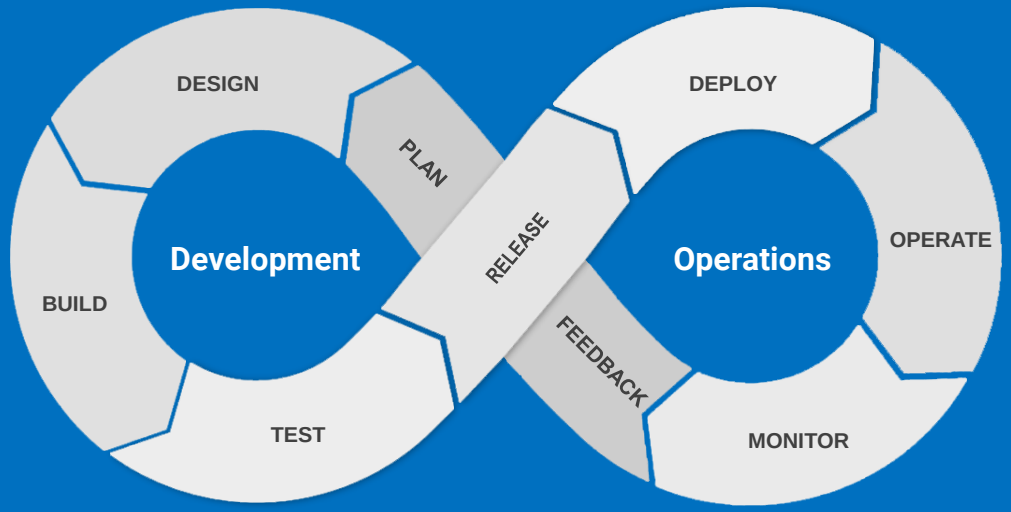
A few Continuous Integration success stories in Automotive

A123 SYSTEMS

A123 link: Similar case study, but uses Jenkins

HL Klemove link: Focus on Polyspace

Summary



Model DevOps

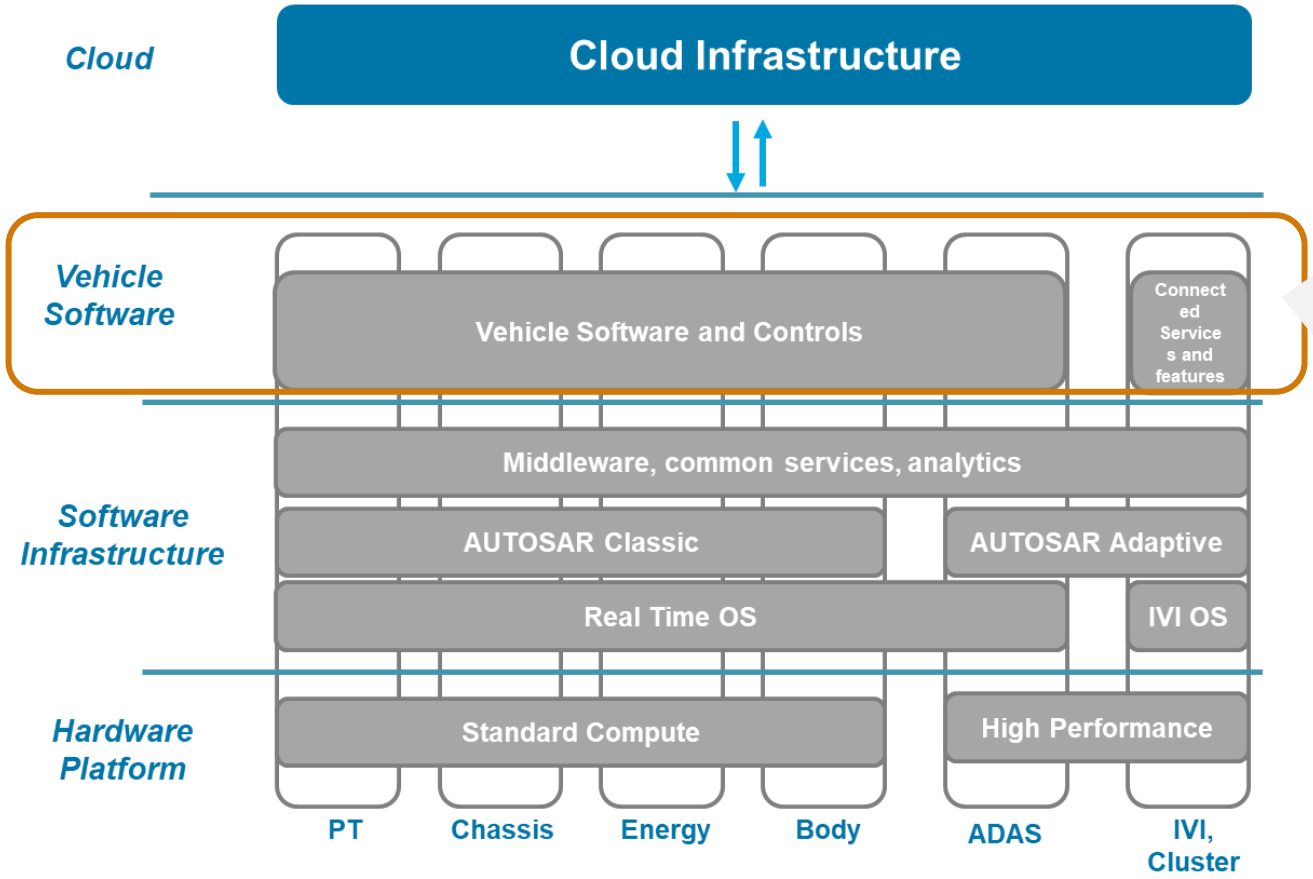
Engineering, Data Science, IT, and Operations teams must collaborate to ensure success

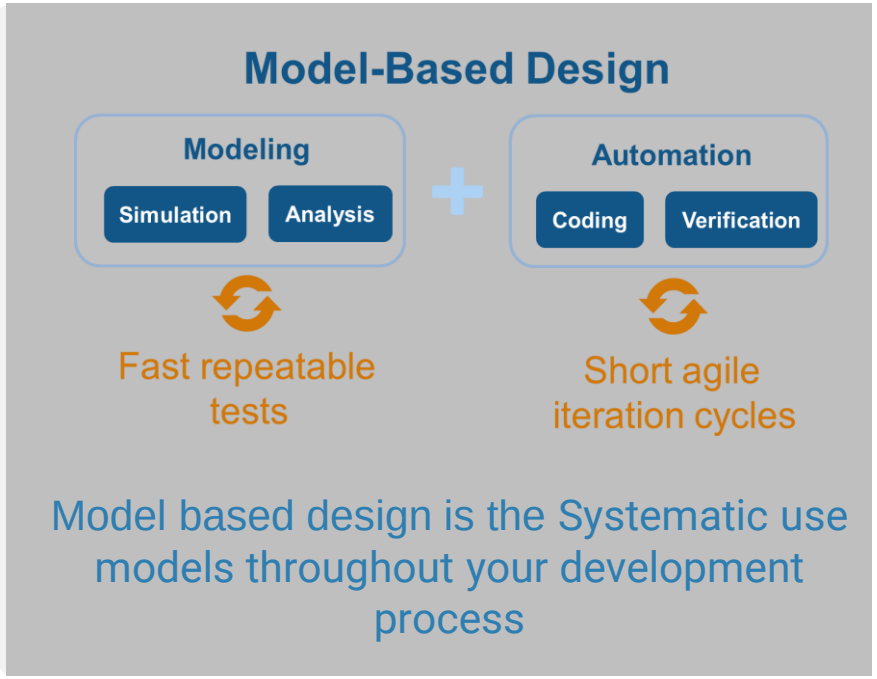
MATLAB & Simulink can be integrated into your development environment and leverage data from a variety of data sources

MATLAB & Simulink models can be deployed into a variety of platforms: embedded, edge, IT/OT, and cloud

Learn more: Continuous Integration for Model-Based Design

Modeling and Automation for Software Defined Vehicle Applications





MathWorks

AUTOMOTIVE CONFERENCE 2023

India

Thank you