Best Practices for Building Large Models from Components to Complex Systems

Erick Saldana Sanvicente, MathWorks

- Works on updating and improving the best practices for large-scale modeling at MathWorks

- 10 years of experience working with embedded systems based on Model-Based Design processes at different automotive OEMs and Tier 1 suppliers.

- Erick holds a B.Sc. in mechatronics.

Brad Hieb, MathWorks

- Focuses on the development and use of large-scale models for automotive system development at MathWorks.

- 14 years of embedded systems design experience at Ford Motor Company

- Brad holds an M.S. in electrical engineering.

MathWorks
AUTOMOTIVE CONFERENCE 2024
North America

Best Practices for Large-Scale Modeling with Simulink

Brad Hieb, MathWorks

(He/Him)

Erick Saldana, MathWorks

(He/Him)

MathWorks®

Introducing Misfire Motors Inc.

Their goal:

To design, simulate and build the most sophisticated vehicle in the world.

Misfiring on all cylinders

Challenge
ARCHITECTURE: Huge monolithic model containing hundreds of thousands of blocks
DATA MANAGEMENT: Unorganized model data living in the base workspace
INTERFACES: Hundreds of non-bus signals connecting virtual subsystems
FILE MANAGEMENT: Ad-hoc file management lacking version control
PERFORMANCE: Simulation running 30x slower than wall clock and taking forever to build

Resolving Misfire Motors modelling issues

Challenge	Best Practice
ARCHITECTURE: Huge monolithic model containing hundreds of thousands of blocks	
DATA MANAGEMENT: Unorganized model data living in the base workspace	
INTERFACES: Hundreds of non-bus signals connecting virtual subsystems	
FILE MANAGEMENT: Ad-hoc file management lacking version control	
PERFORMANCE: Simulation running 30x slower than wall clock and taking forever to build	

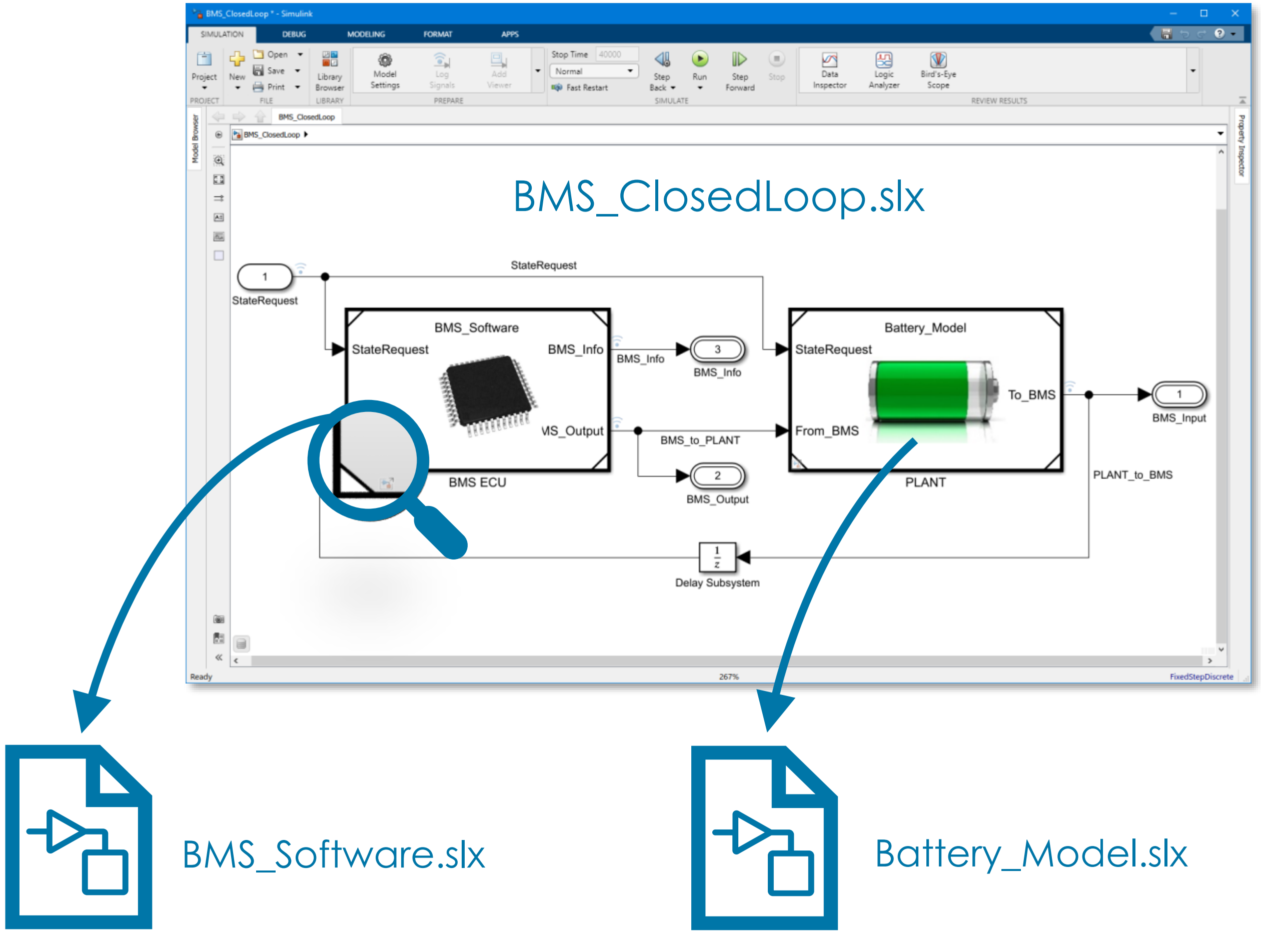
Selecting the right component for each use case

If you want to…	Use..
Visually organize a group of blocks or components	In-Model Virtual Subsystem
Encapsulate small pieces of code in non-reusable packages	In-Model Atomic Subsystem
Create utilities that are widely reused and change infrequently	Linked Subsystems (Libraries)
Create a reusable component that supports physical connection (Simscape) interfaces	(Virtual) Subsystem Reference
Develop/simulate a component as a standalone model	Model Reference
Create a version of your component with IP protection	Model Reference
Use accelerator mode to speed up subsequent instances/runs of a component	Model Reference

Leveraging Simulink’s multitool: model reference

- Simulink lets you develop your components as independent models.
  - Separate .slx, fully functional and simulatable.

- Models can be placed (referenced) inside other models for integration.
- Enables:
  - Reduced build time when reused.
  - Accelerator and Rapid Accelerator modes.



Resolving Misfire Motors modelling issues

Challenge	Best Practice
ARCHITECTURE: Huge monolithic model containing hundreds of thousands of blocks	Componentization, mainly via Model Reference
DATA MANAGEMENT: Unorganized model data living in the base workspace	
INTERFACES: Hundreds of non-bus signals connecting virtual subsystems	
FILE MANAGEMENT: Ad-hoc file management lacking version control	
PERFORMANCE: Simulation running 30x slower than wall clock and taking forever to build	

Scaling up data management with data dictionaries

Base Workspace

- Informal workflows
- Quick parameter tuning
- Single developer
- Universal visibility

Data Dictionary

- Large projects
- Distributed development
- Scoped data

Inherent data persistence

- Dedicated file format (.sldd)

Data organization

- Partition data via referenced dictionaries
- Warning for name clashing

Change tracking workflow

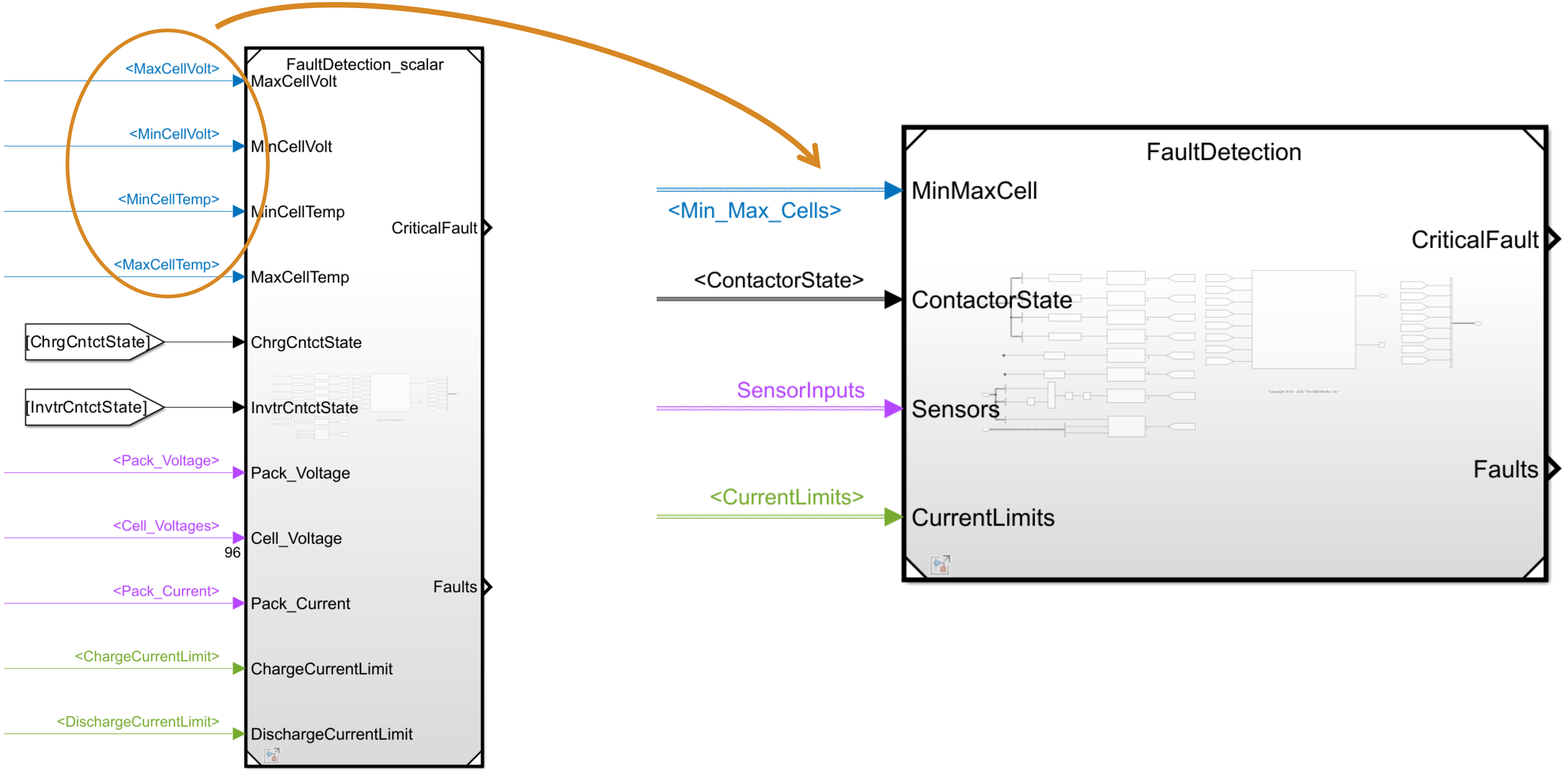
- Changes, last modified time, last modified by
- Show, revert, diff, and save changes

You can slowly migrate from base workspace to data dictionaries as your project scales and then stop accessing base workspace.

Resolving Misfire Motors modelling issues

Challenge	Best Practice
ARCHITECTURE: Huge monolithic model containing hundreds of thousands of blocks	Componentization, mainly via Model Reference
DATA MANAGEMENT: Unorganized model data living in the base workspace	Data distribution, via Simulink Data Dictionaries
INTERFACES: Hundreds of non-bus signals connecting virtual subsystems	
FILE MANAGEMENT: Ad-hoc file management lacking version control	
PERFORMANCE: Simulation running 30x slower than wall clock and taking forever to build	

Simplifying interfaces with buses

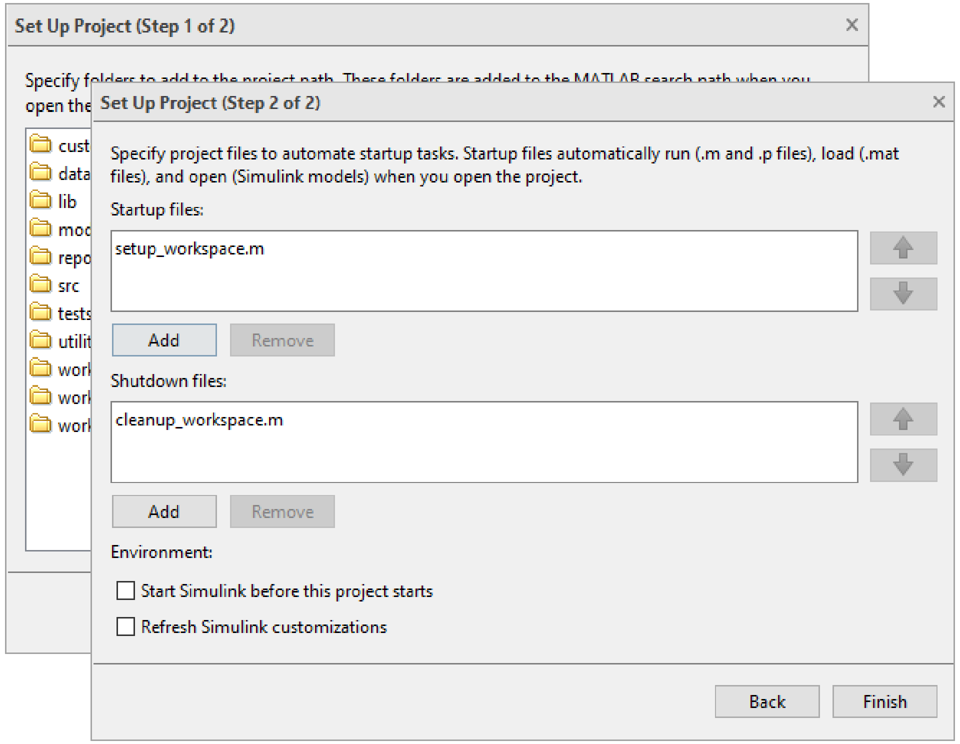


Resolving Misfire Motors modelling issues

Challenge	Best Practice
ARCHITECTURE: Huge monolithic model containing hundreds of thousands of blocks	Componentization, mainly via Model Reference
DATA MANAGEMENT: Unorganized model data living in the base workspace	Data distribution, via Simulink Data Dictionaries
INTERFACES: Hundreds of non-bus signals connecting virtual subsystems	Consolidation in Buses and Bus Element Ports
FILE MANAGEMENT: Ad-hoc file management lacking version control	
PERFORMANCE: Simulation running 30x slower than wall clock and taking forever to build	

Automating the little things, so you have time for the important things

- Sets up the Path
- Configures the Start Up and Shutdown files
- Opens common files at launch
- Creates shortcuts to commonly used tasks



Set Up Project (Step 1 of 2)

Specify folders to add to the project path. These folders are added to the MATLAB search path when you open the

cust
data
lib
mod
repo
src
tests
utilit
work
work
work

Set Up Project (Step 2 of 2)

Specify project files to automate startup tasks. Startup files automatically run (.m and .p files), load (.mat files), and open (Simulink models) when you open the project.

Startup files:

setup_workspace.m

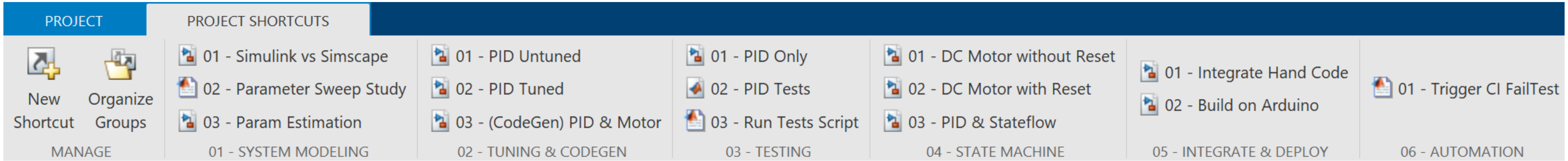
Shutdown files:

cleanup_workspace.m

Environment:

□ Start Simulink before this project starts

□ Refresh Simulink customizations



PROJECT | PROJECT SHORTCUTS

MANAGE: New Shortcut, Organize Groups

01 - SYSTEM MODELING:
01 - Simulink vs Simscape
02 - Parameter Sweep Study
03 - Param Estimation

02 - TUNING & CODEGEN:
01 - PID Untuned
02 - PID Tuned
03 - (CodeGen) PID & Motor

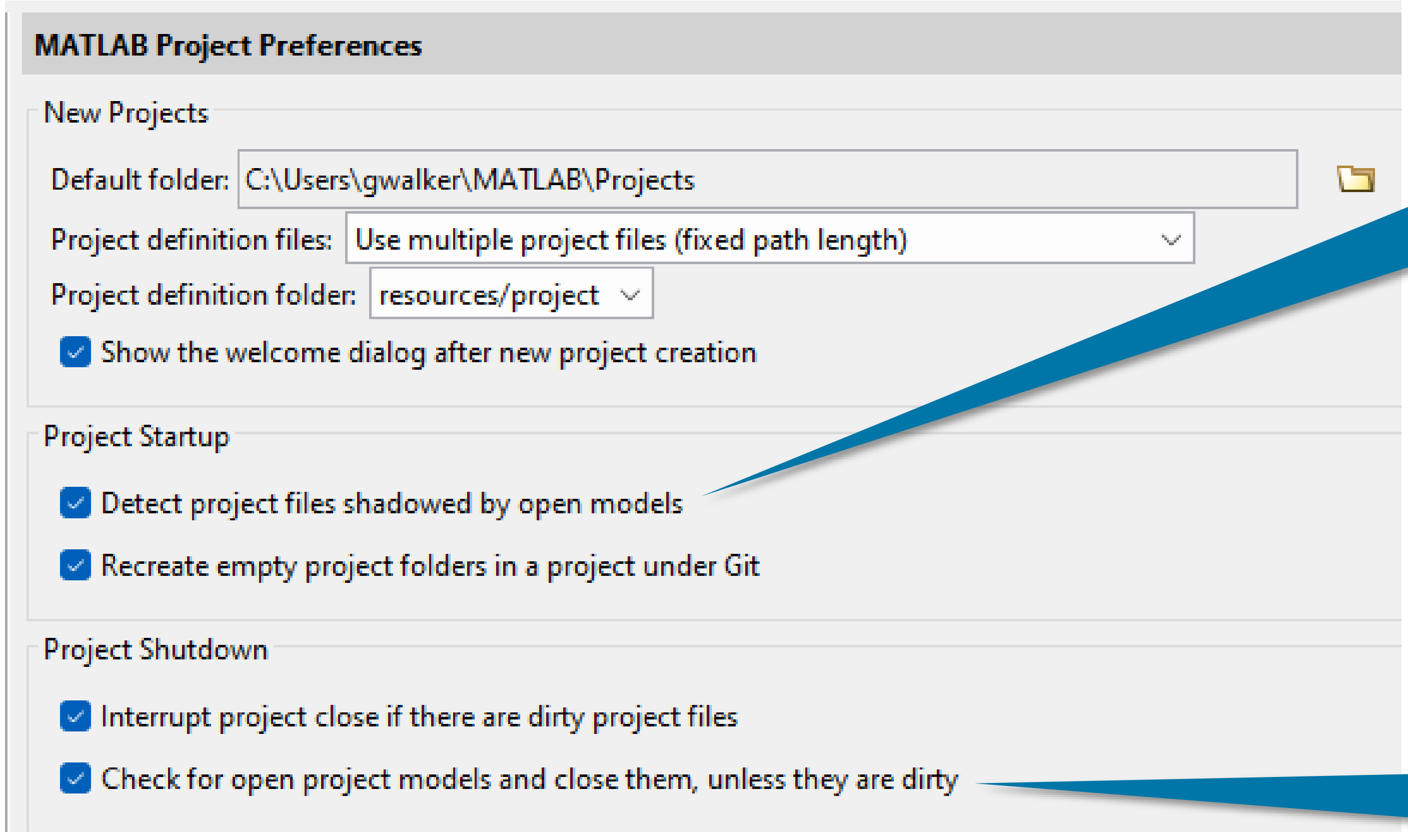
03 - TESTING:
01 - PID Only
02 - PID Tests
03 - Run Tests Script

04 - STATE MACHINE:
01 - DC Motor without Reset
02 - DC Motor with Reset
03 - PID & Stateflow

05 - INTEGRATE & DEPLOY:
01 - Integrate Hand Code
02 - Build on Arduino

06 - AUTOMATION:
01 - Trigger CI FailTest

“Mistake-proofing” your work through projects



MATLAB Project Preferences

New Projects

Default folder: C:\Users\gwalker\MATLAB\Projects

Project definition files: Use multiple project files (fixed path length)

Project definition folder: resources/project

☑ Show the welcome dialog after new project creation

Project Startup

☑ Detect project files shadowed by open models

☑ Recreate empty project folders in a project under Git

Project Shutdown

☑ Interrupt project close if there are dirty project files

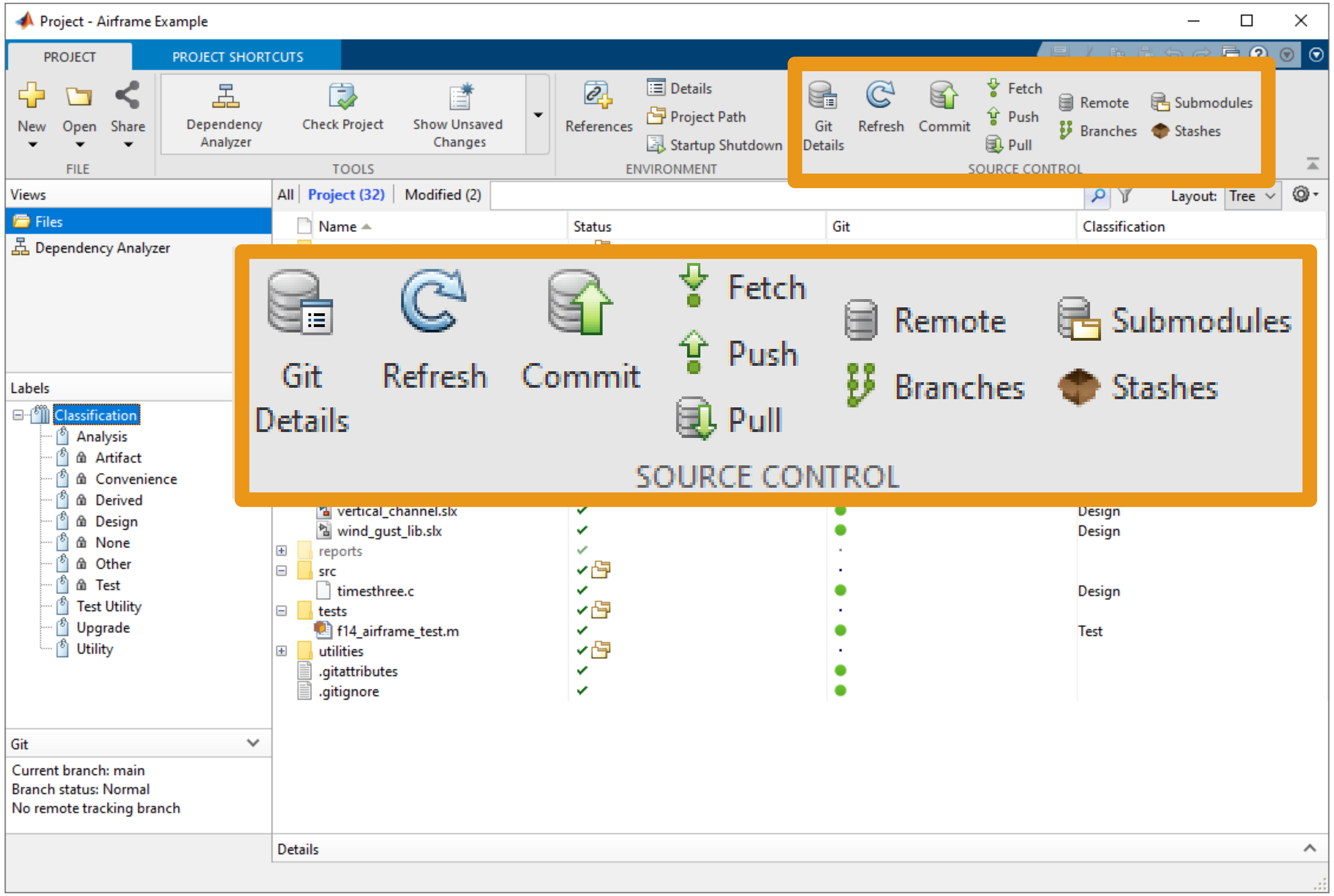
☑ Check for open project models and close them, unless they are dirty

Shadowed models & libraries can give unexpected results: the project finds them and warns you

Unsaved changes risk losing work or getting confusing results: the project will ask you to confirm

Managing source control with projects

- Refresh
- Pull
- Commit
- Push
- Branches (history)



Name	Status	Git	Classification
vertical_channel.slx	✓	●	Design
wind_gust_lib.slx	✓	●	Design
reports	✓	·	
src	✓	·	
timesthree.c	✓	●	Design
tests	✓	·	
f14_airframe_test.m	✓	●	Test
utilities	✓	·	
.gitattributes	✓	●	
.gitignore	✓	●	

Current branch: main
Branch status: Normal
No remote tracking branch

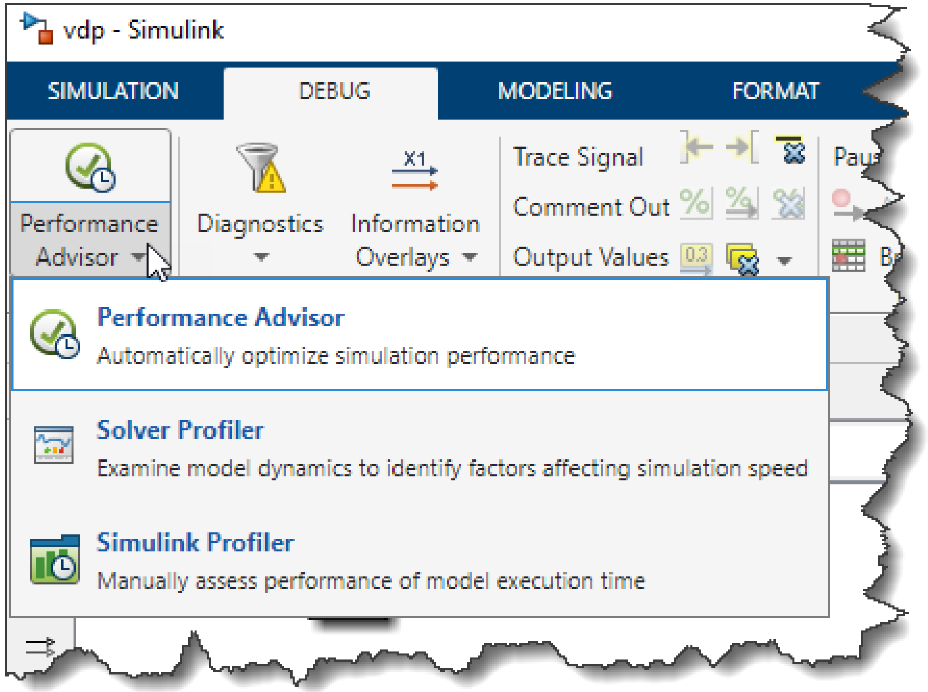
Resolving Misfire Motors modelling issues

Challenge	Best Practice
ARCHITECTURE: Huge monolithic model containing hundreds of thousands of blocks	Componentization, mainly via Model Reference
DATA MANAGEMENT: Unorganized model data living in the base workspace	Data distribution, via Simulink Data Dictionaries
INTERFACES: Hundreds of non-bus signals connecting virtual subsystems	Consolidation in Buses and Bus Element Ports
FILE MANAGEMENT: Ad-hoc file management lacking version control	Organization with MATLAB Projects
PERFORMANCE: Simulation running 30x slower than wall clock and taking forever to build	

Optimizing performance

Optimizing performance

Tool	Typical use case
Performance Advisor	First step when diagnosing performance issues
MATLAB Profiler	Initialization profiling
Simulink Profiler	Simulation profiling
Solver Profiler	Simulation profiling when using variable step solver

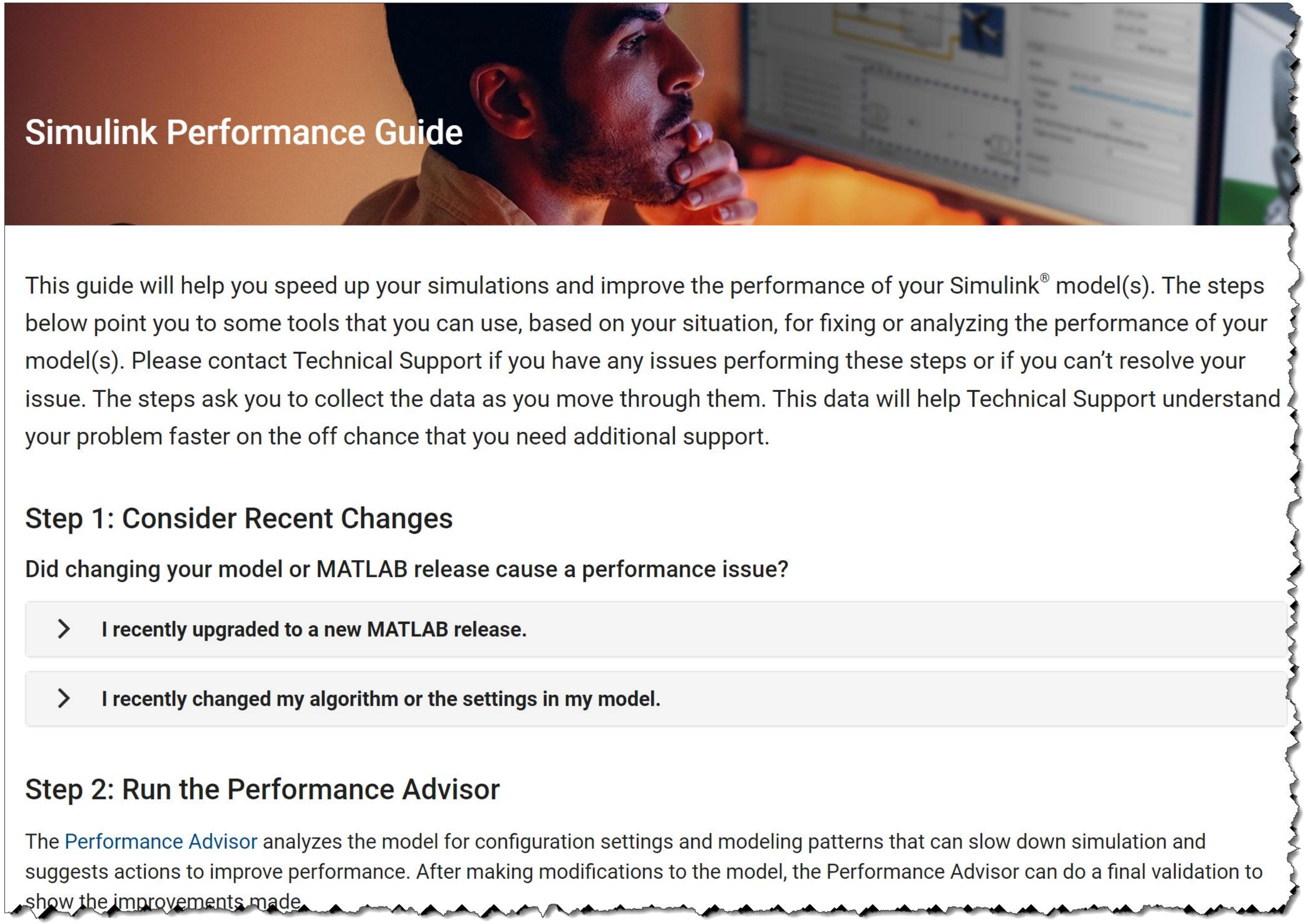


Performance Advisor
Automatically optimize simulation performance

Solver Profiler
Examine model dynamics to identify factors affecting simulation speed

Simulink Profiler
Manually assess performance of model execution time

Introducing the Simulink Performance Guide



Simulink Performance Guide

This guide will help you speed up your simulations and improve the performance of your Simulink® model(s). The steps below point you to some tools that you can use, based on your situation, for fixing or analyzing the performance of your model(s). Please contact Technical Support if you have any issues performing these steps or if you can't resolve your issue. The steps ask you to collect the data as you move through them. This data will help Technical Support understand your problem faster on the off chance that you need additional support.

Step 1: Consider Recent Changes

Did changing your model or MATLAB release cause a performance issue?

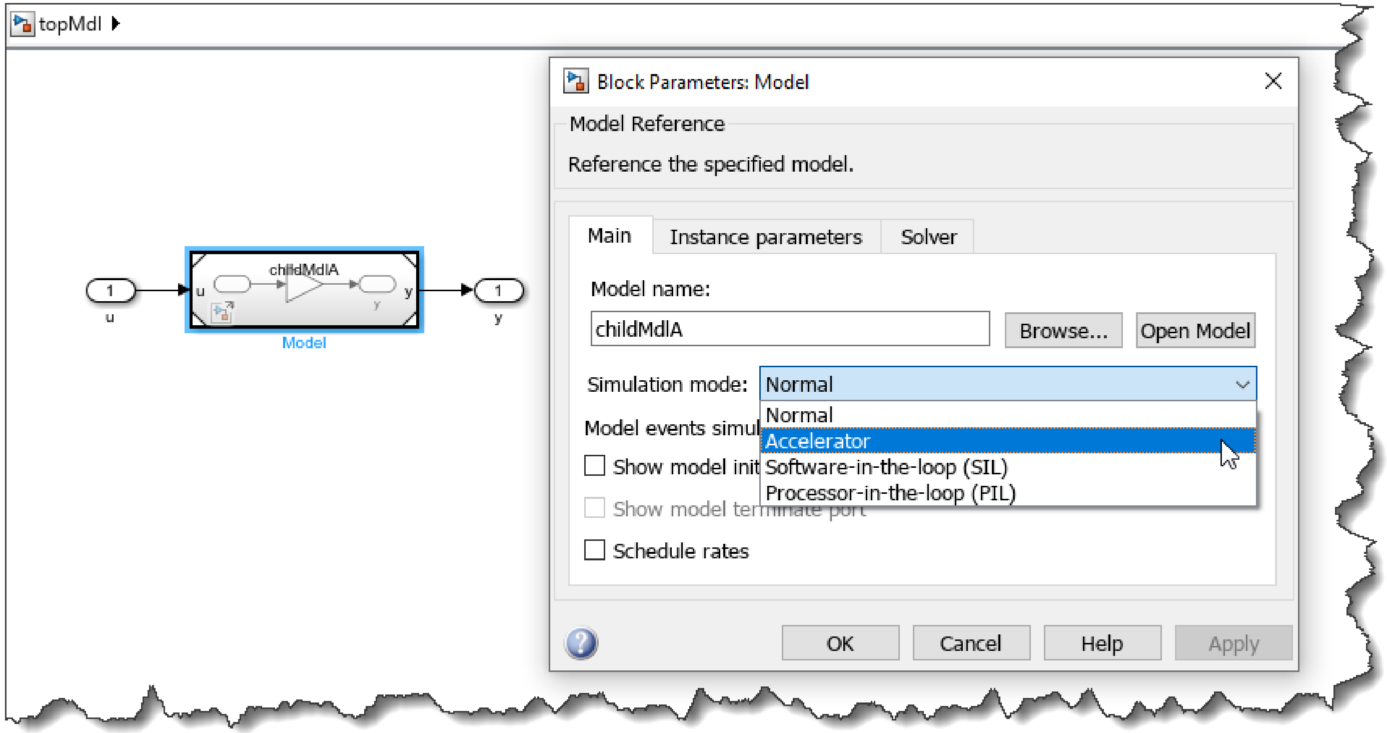
> **I recently upgraded to a new MATLAB release.**

> **I recently changed my algorithm or the settings in my model.**

Step 2: Run the Performance Advisor

The Performance Advisor analyzes the model for configuration settings and modeling patterns that can slow down simulation and suggests actions to improve performance. After making modifications to the model, the Performance Advisor can do a final validation to show the improvements made.

Speeding up simulation with Accelerator Mode



Block Parameters: Model

Model Reference

Reference the specified model.

Main | Instance parameters | Solver

Model name: childMdlA Browse... Open Model

Simulation mode: Normal
- Normal
- Accelerator
- Software-in-the-loop (SIL)
- Processor-in-the-loop (PIL)

Model events simul

□ Show model init

□ Show model terminate port

□ Schedule rates

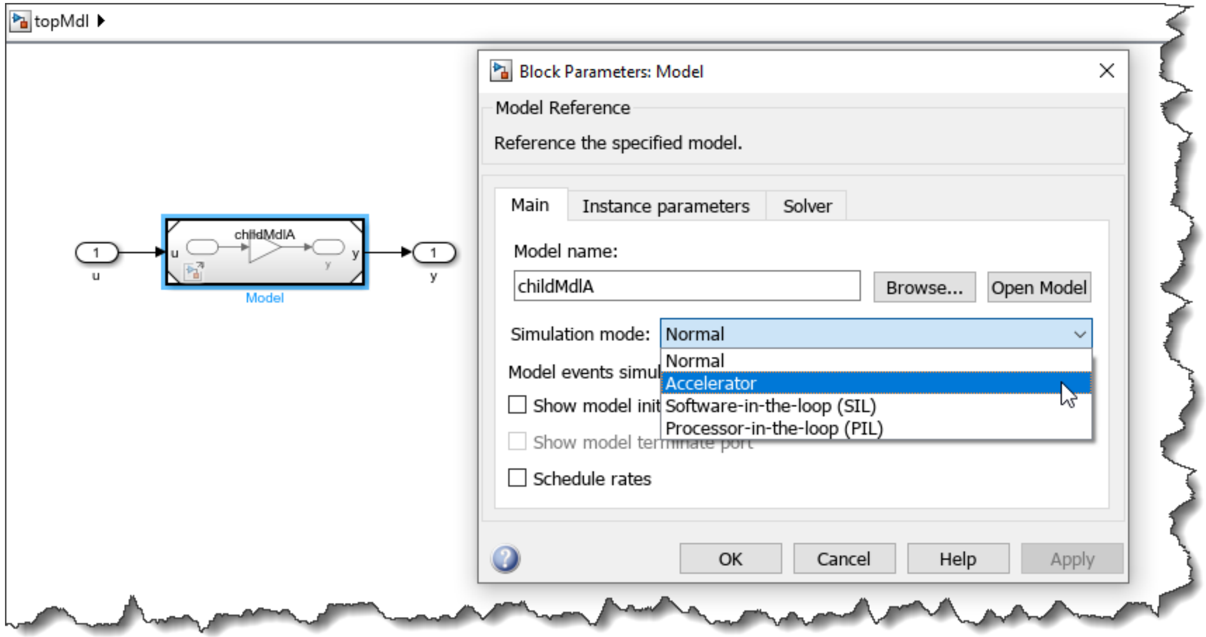
OK Cancel Help Apply

In **Normal** mode the model is executed in interpreted mode.

In **Accelerator** mode the model is converted to C code and compiled:

- Faster simulation
- Much shorter initialization time

Speeding up initialization with incremental builds



Model Reference

Reference the specified model.

Model name: childMdlA

Simulation mode: Normal

Model events simul

☐ Show model init

☐ Show model terminate port

☐ Schedule rates

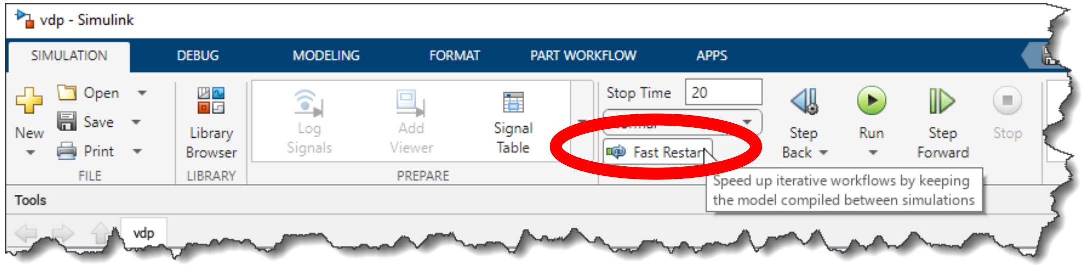
- Simulink Cache (.slxc) file
  - Stores pre-compiled model component
- Model Initialization
  - Simulink loads previously pre-compiled component models from Cache files
  - Simulink only rebuilds compiled component models that are out of date

“Incremental build”

Results in much shorter initialization time

Reducing initialization time with Fast Restart

- Useful when simulating a model without structural changes between runs
- First simulation compiles the model
- Model remains compiled for next simulations

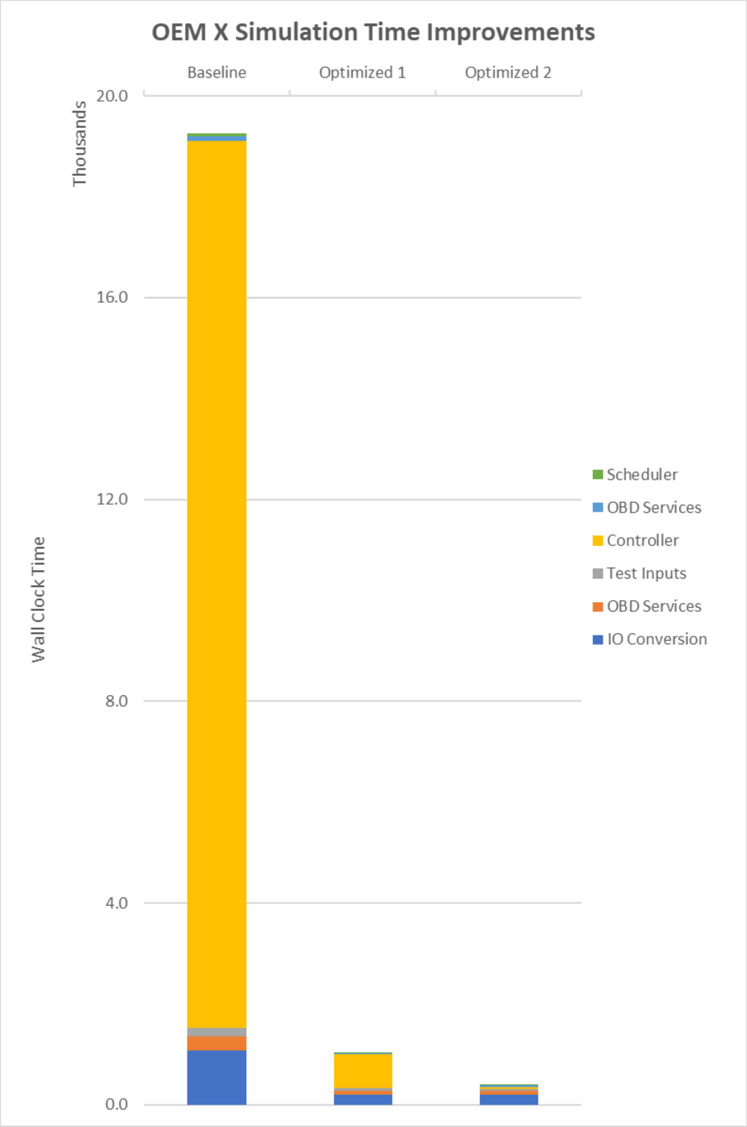


Speed up iterative workflows by keeping the model compiled between simulations

```
modelname = 'vdp';
in(1:2) = Simulink.SimulationInput(modelname);
out = sim(in,'UseFastRestart','on');
t1 = out(1).SimulationMetadata.TimingInfo.InitializationElapsedWallTime
t1 =
    5.6549

t2 = out(2).SimulationMetadata.TimingInfo.InitializationElapsedWallTime
t2 =
    0.0380
```

Performing a real-world performance investigation



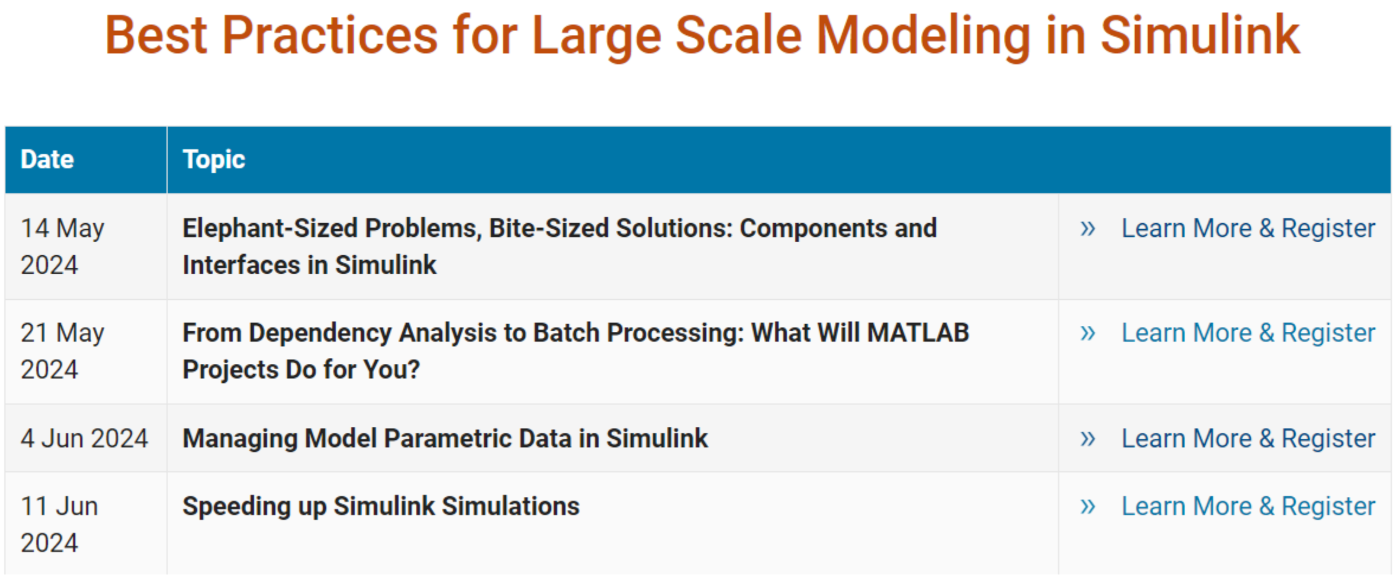
- This investigation focused on simulation speed
- Used performance analysis tools to identify issues
- Applied several corrective actions including:
  - Accelerator mode
  - Model optimization
- Overall simulation performance improved by 67x

Resolving Misfire Motors modelling issues

Challenge	Best Practice
ARCHITECTURE: Huge monolithic model containing hundreds of thousands of blocks	Componentization, mainly via Model Reference
DATA MANAGEMENT: Unorganized model data living in the base workspace	Data distribution, via Simulink Data Dictionaries
INTERFACES: Hundreds of non-bus signals connecting virtual subsystems	Consolidation in Buses and Bus Element Ports
FILE MANAGEMENT: Ad-hoc file management lacking version control	Organization with MATLAB Projects
PERFORMANCE: Simulation running 30x slower than wall clock and taking forever to build	Evaluation and improvement through dedicated tools

Extending your knowledge with dedicated webinar series

FREE 4-part series about
Large-Scale Modeling



Best Practices for Large Scale Modeling in Simulink

Date	Topic	
14 May 2024	**Elephant-Sized Problems, Bite-Sized Solutions: Components and Interfaces in Simulink**	» Learn More & Register
21 May 2024	**From Dependency Analysis to Batch Processing: What Will MATLAB Projects Do for You?**	» Learn More & Register
4 Jun 2024	**Managing Model Parametric Data in Simulink**	» Learn More & Register
11 Jun 2024	**Speeding up Simulink Simulations**	» Learn More & Register

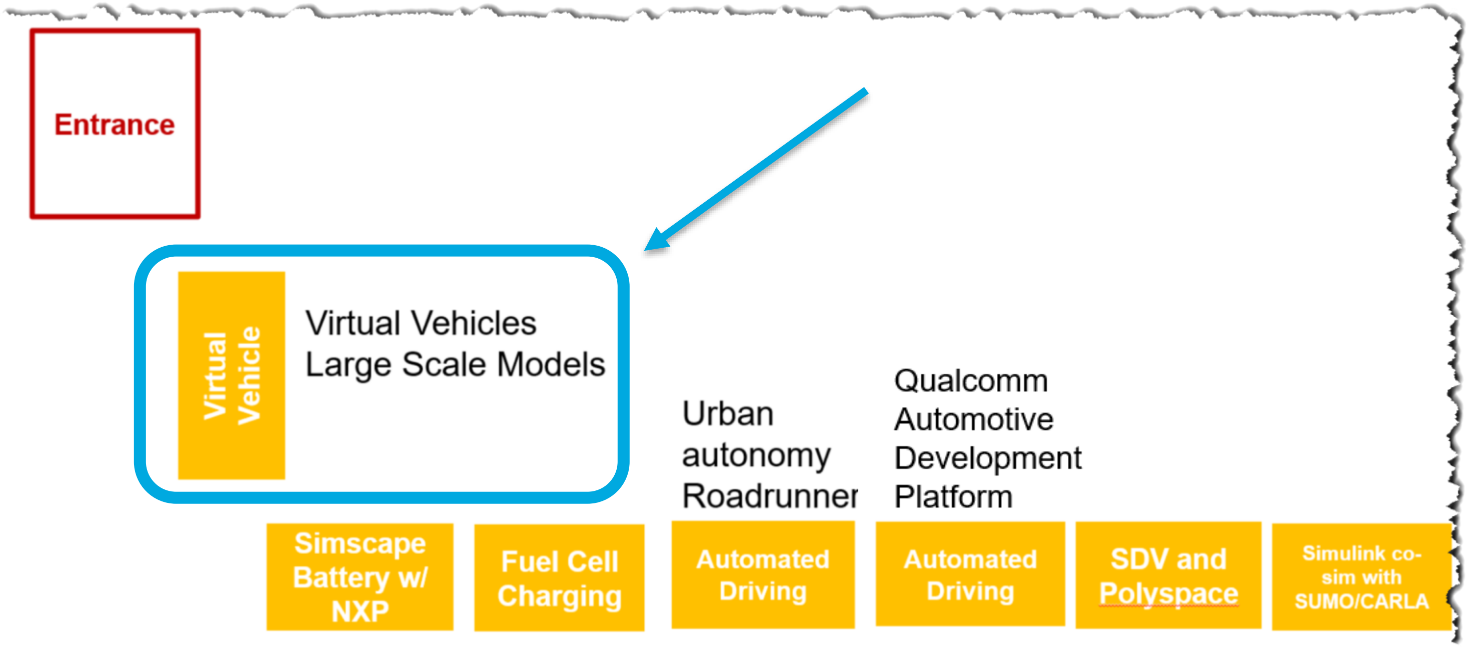
Learning more about these best practices

These slides

Passcode: MACNA2024

Come talk to us!

We have a booth outside



Entrance

Virtual Vehicle

Virtual Vehicles
Large Scale Models

Urban autonomy Roadrunner

Qualcomm Automotive Development Platform

Simscape Battery w/ NXP

Fuel Cell Charging

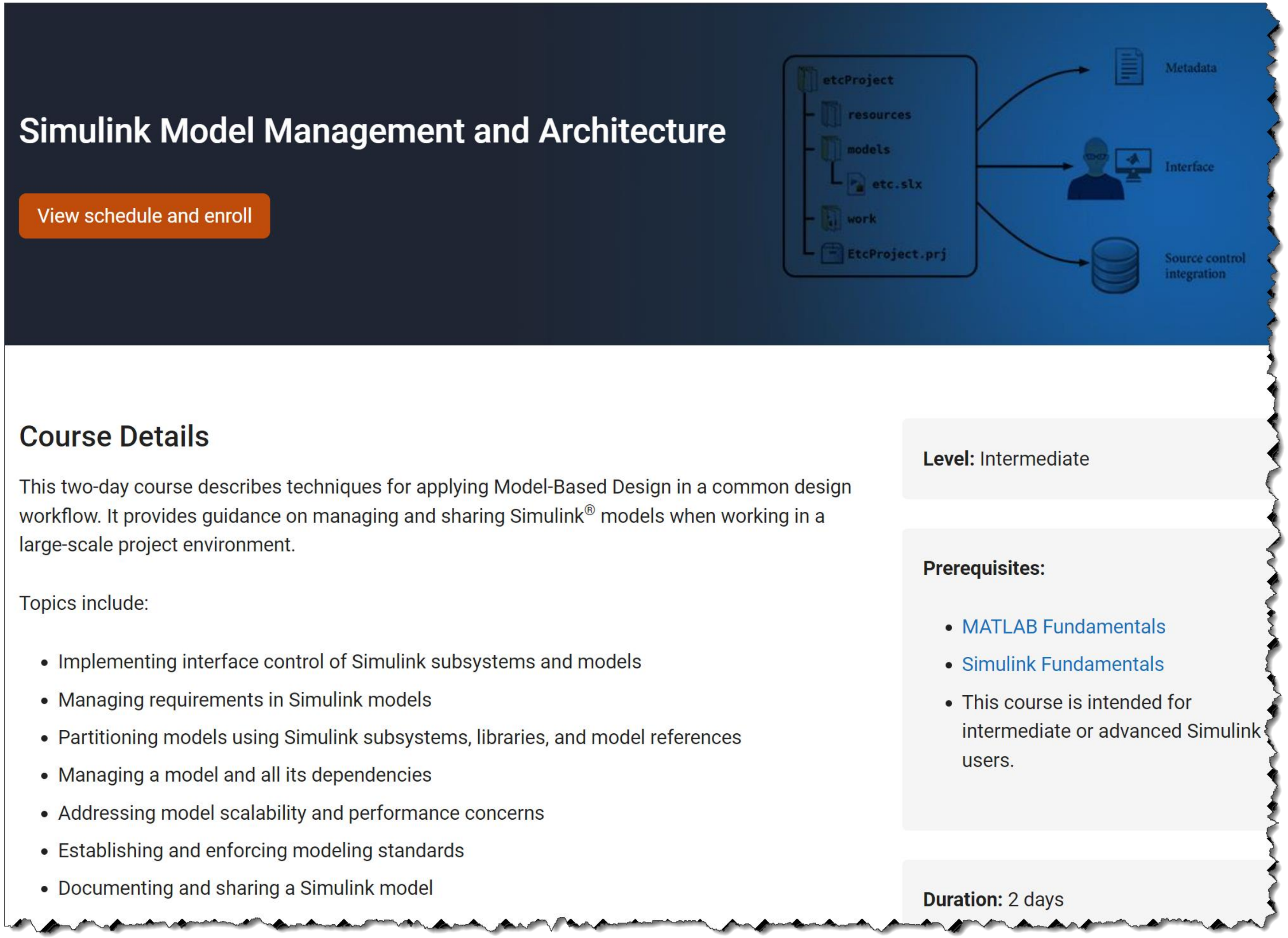
Automated Driving

Automated Driving

SDV and Polyspace

Simulink co-sim with SUMO/CARLA

Reinforcing your knowledge with advanced training



Simulink Model Management and Architecture

Course Details

This two-day course describes techniques for applying Model-Based Design in a common design workflow. It provides guidance on managing and sharing Simulink® models when working in a large-scale project environment.

Topics include:

- Implementing interface control of Simulink subsystems and models
- Managing requirements in Simulink models
- Partitioning models using Simulink subsystems, libraries, and model references
- Managing a model and all its dependencies
- Addressing model scalability and performance concerns
- Establishing and enforcing modeling standards
- Documenting and sharing a Simulink model

Level: Intermediate

Prerequisites:

- MATLAB Fundamentals
- Simulink Fundamentals
- This course is intended for intermediate or advanced Simulink users.

Duration: 2 days

2-day training course on Model Management and Architecture

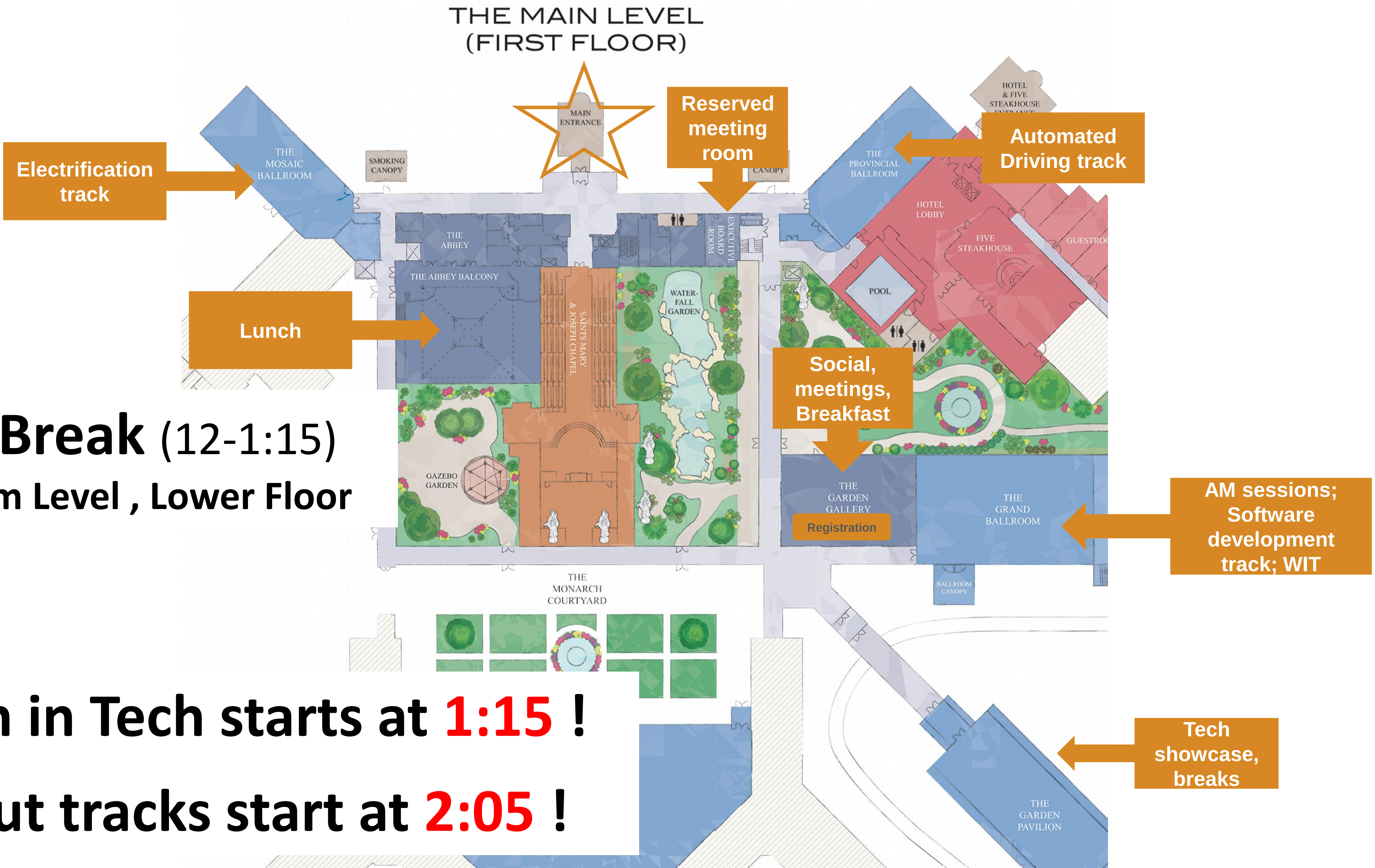
Avoiding Misfire Motors’ mistakes

- Best practices are available to you.
  - Through these slides and webinars.
- Training, foundational and advanced is available.
- MathWorks has Application Engineering, Tech Support and Consulting resources.
- Go into world and preach the Best Practices.

MathWorks
AUTOMOTIVE CONFERENCE 2024
North America

Thank you

MathWorks®



Electrification track

Reserved meeting room

Automated Driving track

Lunch

Social, meetings, Breakfast

Registration

AM sessions; Software development track; WIT

Tech showcase, breaks

Lunch Break (12-1:15)

The Atrium Level , Lower Floor

Women in Tech starts at 1:15 !

Breakout tracks start at 2:05 !