

MathWorks
**AUTOMOTIVE
CONFERENCE 2024**
North America

Migration of Monolithic Algorithm to Service Oriented Architecture

Mark Danielsen, MathWorks

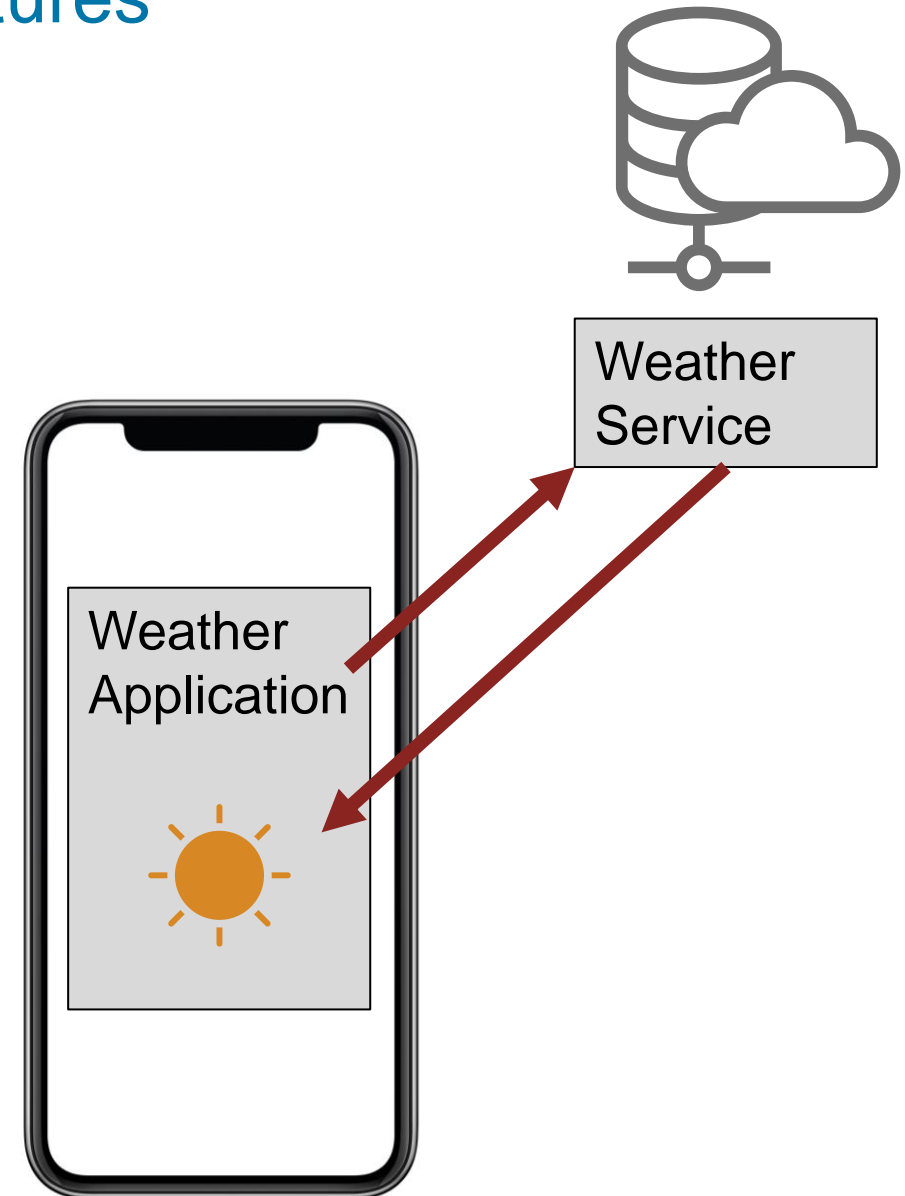


An Example of Service Oriented Architectures

Let's Talk About the Weather ...

Mobile Weather Reports

- Weather App calls out to a Weather Service in the cloud
- Weather Service gets request and responds with a block a data that represents current weather info
- Weather App decodes and displays the information



An Example of Service Oriented Architectures

Let's Talk About the Weather ...

Mobile Weather Reports

- Weather App calls out to a Weather Service in the cloud
- Weather Service gets request and responds with a block a data that represents current weather info
- Weather App decodes and displays the information

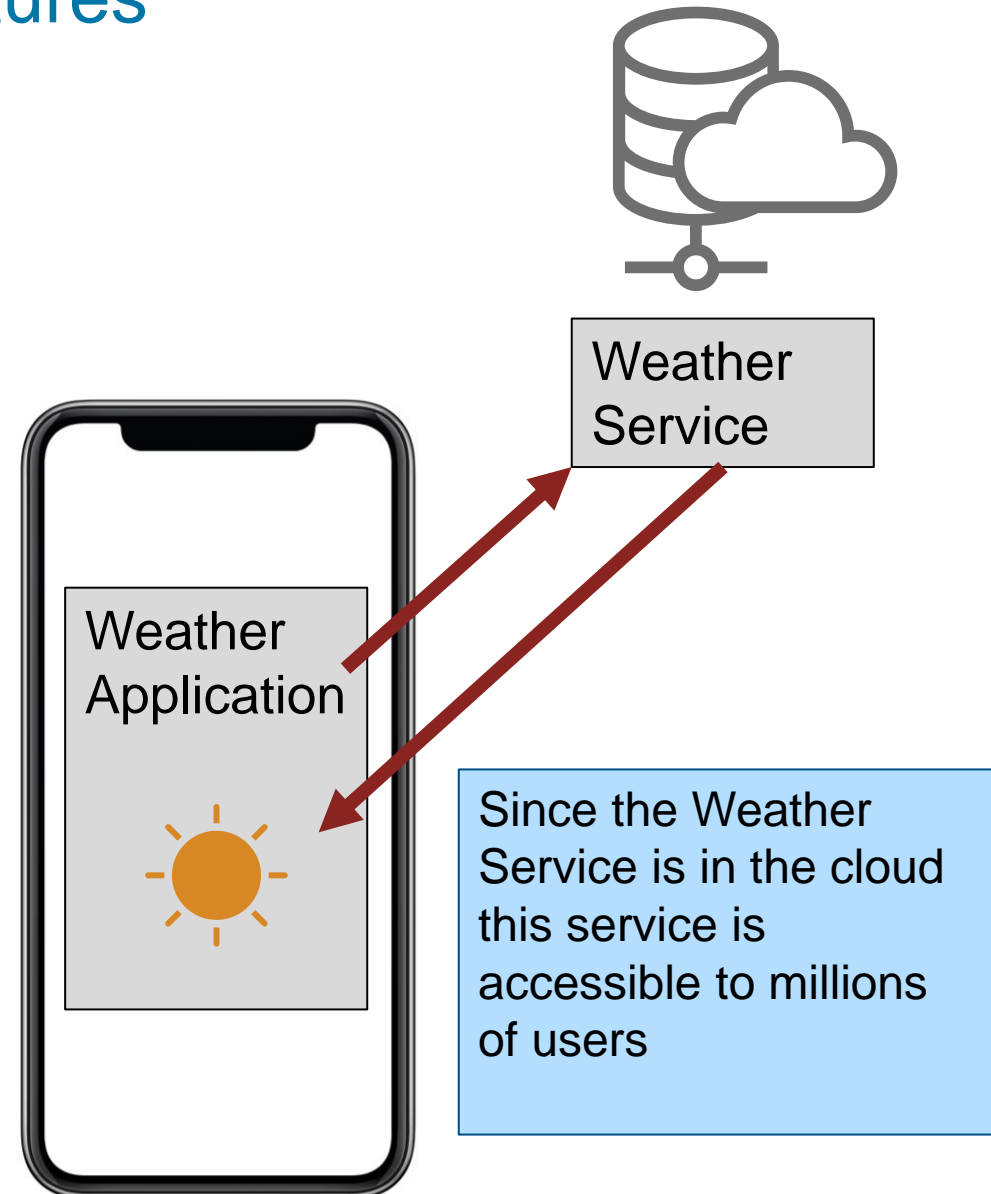
More Details:

Weather Application

Calling out to a function that is not part of the Weather App

Weather Service is providing some function or service

The Service may or may not respond depending on the function or application



Key Take Aways

- We have the tools to help move to Service Oriented Architectures (SOA)
- We can generate C++ code for SOA based components
- We are going to show you a lot of information in this presentation
- We are here to help so please engage with us in your SOA type projects

Challenge Statement and Project Benefits

Challenge:

- How to partition an algorithm into Services that allow for reusability, portability and still maintain functionality of the algorithm
- Repartitioning algorithm content into smaller pieces will consist of engineering design choices and requires engineering rigor

We will show:

- Once the partitioning decisions have been made, how to create the Service Oriented Architecture framework
- How to create interfaces to Service Components
- How to create Simulink behavioral models
- how to create agnostic C++ code
- how to apply Adaptive AUTOSAR

Previous projects that helped defined SOA (Service Oriented Architecture) with Models

We will build from these previous projects to show how to migrate a Monolithic Model to SOA

[Technical Article: Migrating Traditional Automotive Applications to SOA](#)



ANNIVERSARY 2003 - 2023

Migrating traditional automotive application compositions to AUTOSAR Adaptive services for Software Defined Vehicles



Shwetha Bhadravathi Patil, Product Manager

Nukul Sehgal, Senior Application Engineer

May 12, 2023

San Diego



[Applying Model Based Design to SDV Development](#)

MathWorks
**AUTOMOTIVE
CONFERENCE 2023**

Applications for the Software-Defined Vehicle:
Scaling software development with virtual analysis, automated testing, and cloud resources

Sameer K. Muckatira, MathWorks



Nitish Rao, MathWorks



MathWorks

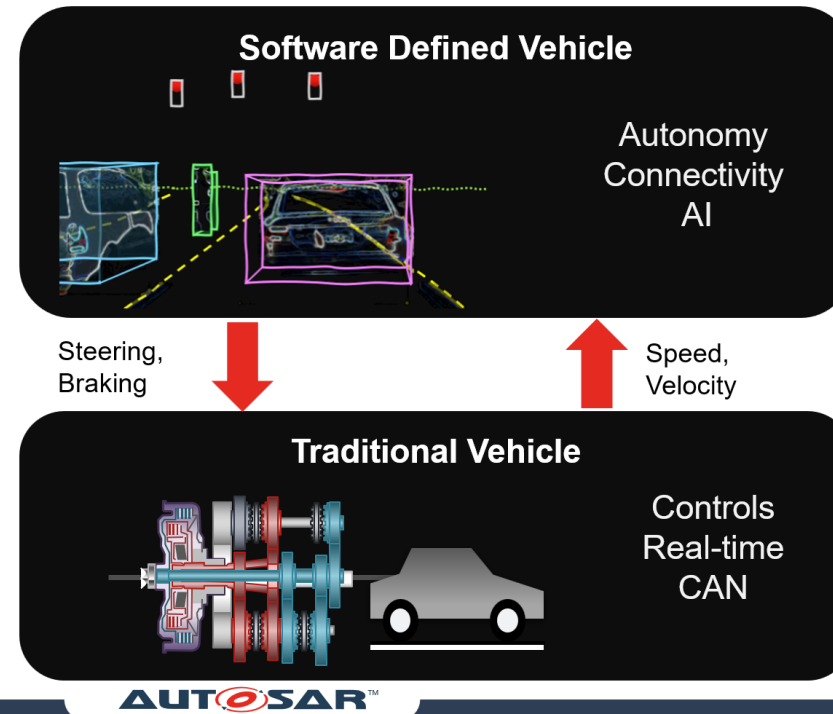
Trend: Automakers are embracing Software Oriented Architectures and Standards

What makes SOA so attractive to Automakers ??

By creating Service Components:

- Reusable Services that may be shared with many applications or components
- Potential for relocation of function
- Higher level of testability
- Potential for reduction in validation due to sharing of functions across application or components

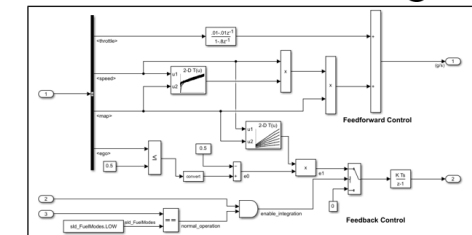
Automakers are increasingly building software in-house with SOA based design



SOA based standards



Model-Based Design



Process of Decomposition to Composition

This is the process of pulling your legacy model apart ...

and building in service components with service interfaces...

to create an SOA framework

How to decompose traditional application software compositions into services for Software Defined Vehicles applications?



Reference to Potential Guiding Principles

8.5.2.1 Single-responsibility Principle

The single-responsibility principle (SRP, SWEBOOK3) [7] states that a component or class should be responsible for a single part of the overall functionality provided by the software. That responsibility should be encapsulated by the software interface provided by one or more interface(s). This principle is a key guiding principle for software design.

The single-responsibility principle (SRP) has several reasons (i.e. reasons) that require a change in design to have a positive impact on code quality and reduce the need for more maintenance.

We will be working on partitioning the application into services.

Requires Engineering Decisions and Choices

We will show you the Engineering Decisions that MathWorks made in order to separate out Service Components

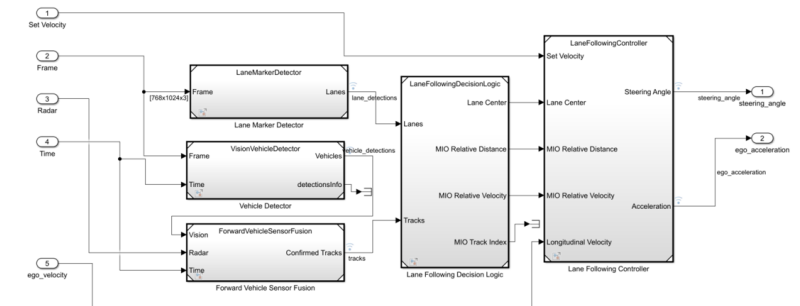
Decompose the Monolithic Application



Some techniques can be found under Section 8.3 Decomposition Strategy and Section 8.5 Design Principles in the standard document - [Explanation of Adaptive Platform Software Architecture R22-11](#)

Few of the notable techniques from design principles

1. Single-responsibility Principle
2. Interface Segregation Principle
3. Dependency Inversion Principle
4. Acyclic Dependencies Principle



A Highway Lane Following application designed previously as a monolithic application composition can be converted to a single service or it can be decomposed into multiple services like camera service, vision service, radar service, lane detection service etc.



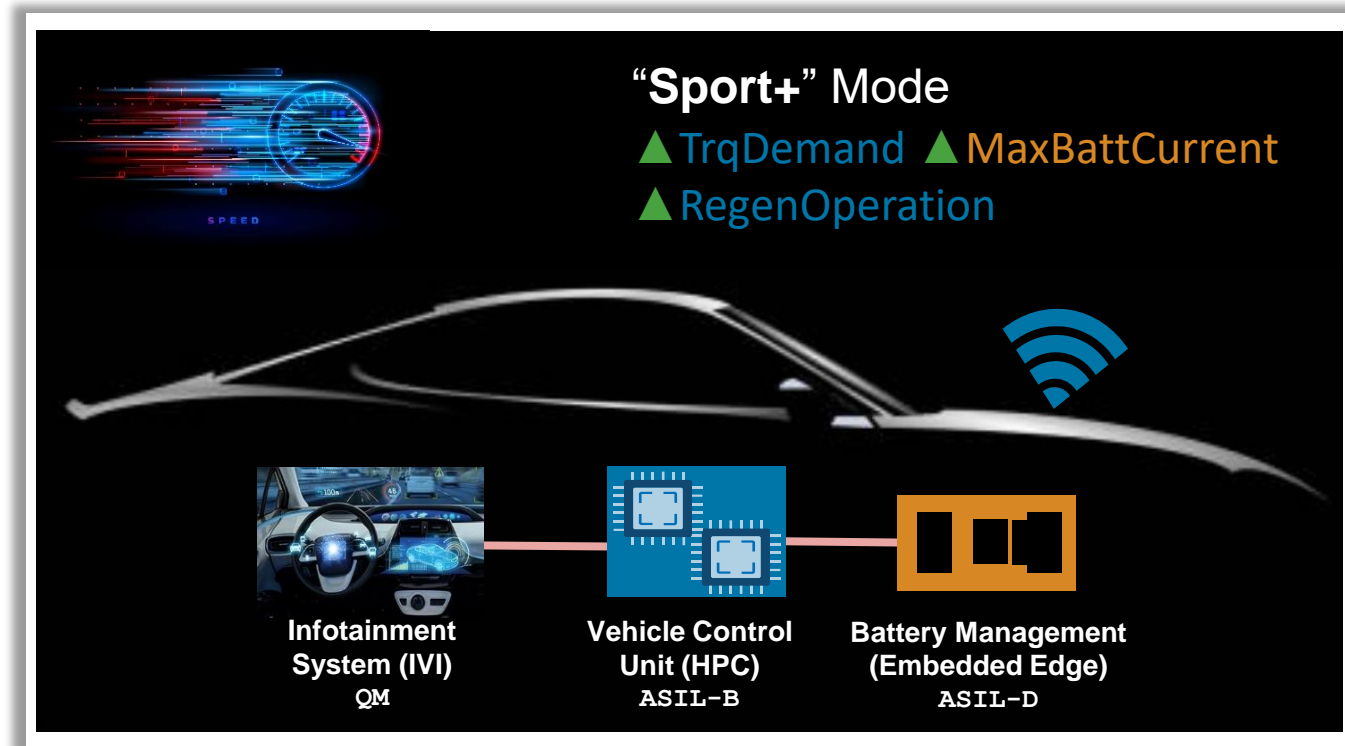
The Models that we are going to work on come from the second presentation that I referenced

This project had models for two different ECUs working together

Battery Management – which consisted of AUTOSAR Classic components for Battery State Health and State of Charge

Vehicle Control Unit (HPC) – Which consisted of algorithms that are slated for an Adaptive Platform

We are going to use the models for the Vehicle Control Unit (HPC) for this demonstration



Start with a Baseline Model with tests that pass

TESTS

Test Browser Results and Artifacts

Filter results by name or tags, e.g. tags: test

NAME	STATUS
Results: 2024-Apr-08 12:30:44	2 ✔
EV2M_VCU_MiTests	2 ✔
VCU_2EMEV_ctrl_powertrain	2 ✔
VCU_2EMEV_Harness_Bas	✔
VCU_2EMEV_Harness_High	✔

PROPERTY	VALUE
Name	Results: 2024-Apr-08 12:30:44
Status	2 ✔
Start Time	04/08/2024 12:30:44
End Time	04/08/2024 12:31:44

The Simulink model diagram shows a complex control system for an electric vehicle powertrain. Key components include:

- Inputs:** AccelPd (1), TransGear (5), VehSpdFdbk (3), BouBrkPrsCmd (2), EMSpd (4), BattPackVolt (7), BMSDischrgCurrLmt (8), BMSChrgCurrLmt (9), SOC (6), DriveMode (13).
- Control Logic:** Round, AccelPd, Gear, VehSpd, Accel Pedal to Traction Wheel Torque Request, BrkPrsReq, Park, Brake Pedal to Total Braking Pressure Request, Goto1, EMSpd, PackVolt, BattPwrDischrgLmt, BattPwrChrgLmt, Power Limit Calculations, SOC Calculation, driverModeChecked, driverMode, MaxDchrgCurrLim.
- Outputs:** VehSpd, BrkPrsReq, EMTrqCmd, BrkCmd, BattStateRequest, BattStateRequest.

Based on Model Functions decide what will be the main app and what will be Services

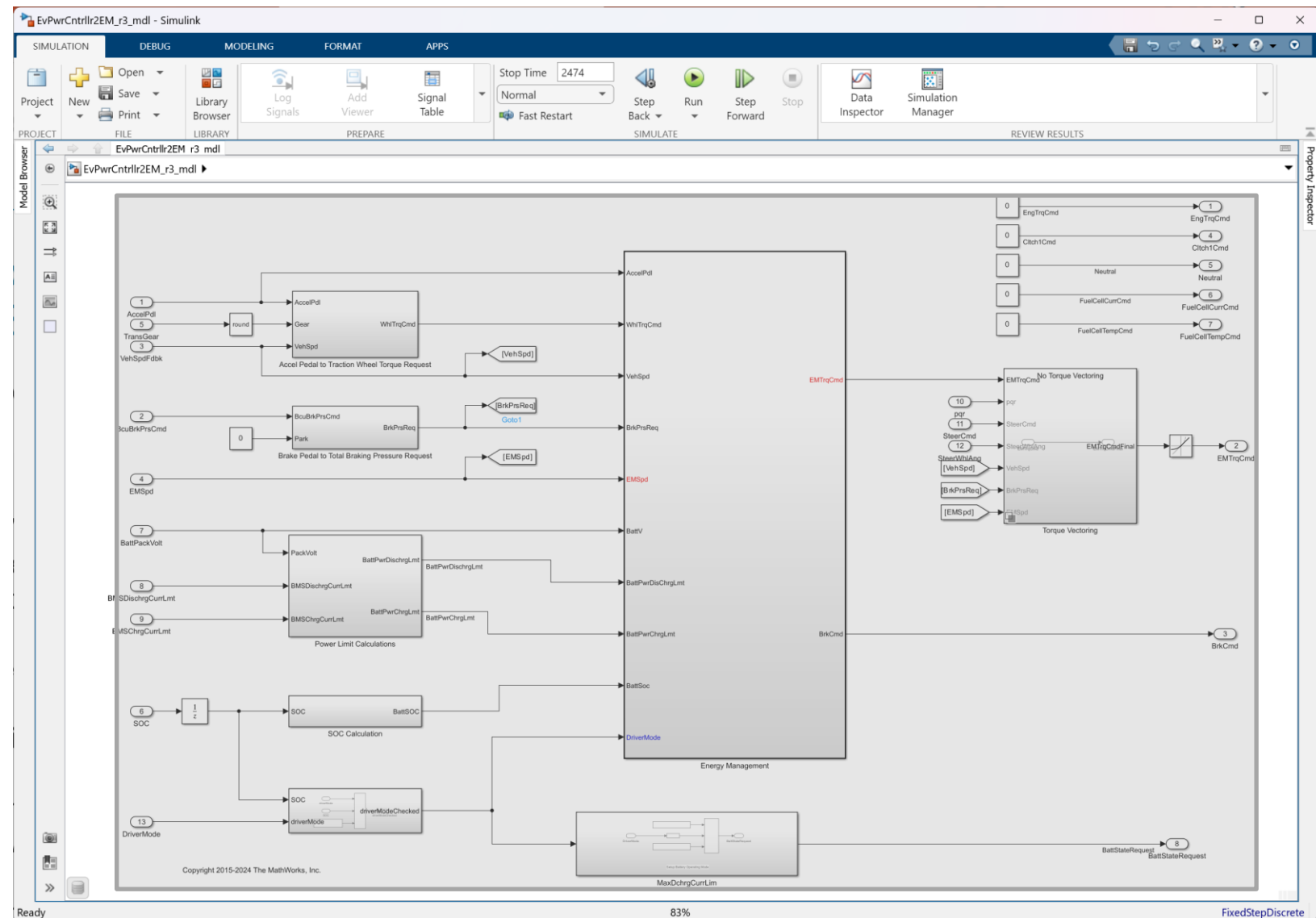
MathWorks Design Decisions

Identify and Analyze Services

Main app will retain full functionality

Our end model will look like the grey box in the model

However, the end refactored model will call out to services in place of some of the inline functionality



Identify and Analyze Services

Define Services and its interfaces

Define Service Contracts

Implement and deploy Services

Based on Model Functions decide what will be the main app and what will be Services

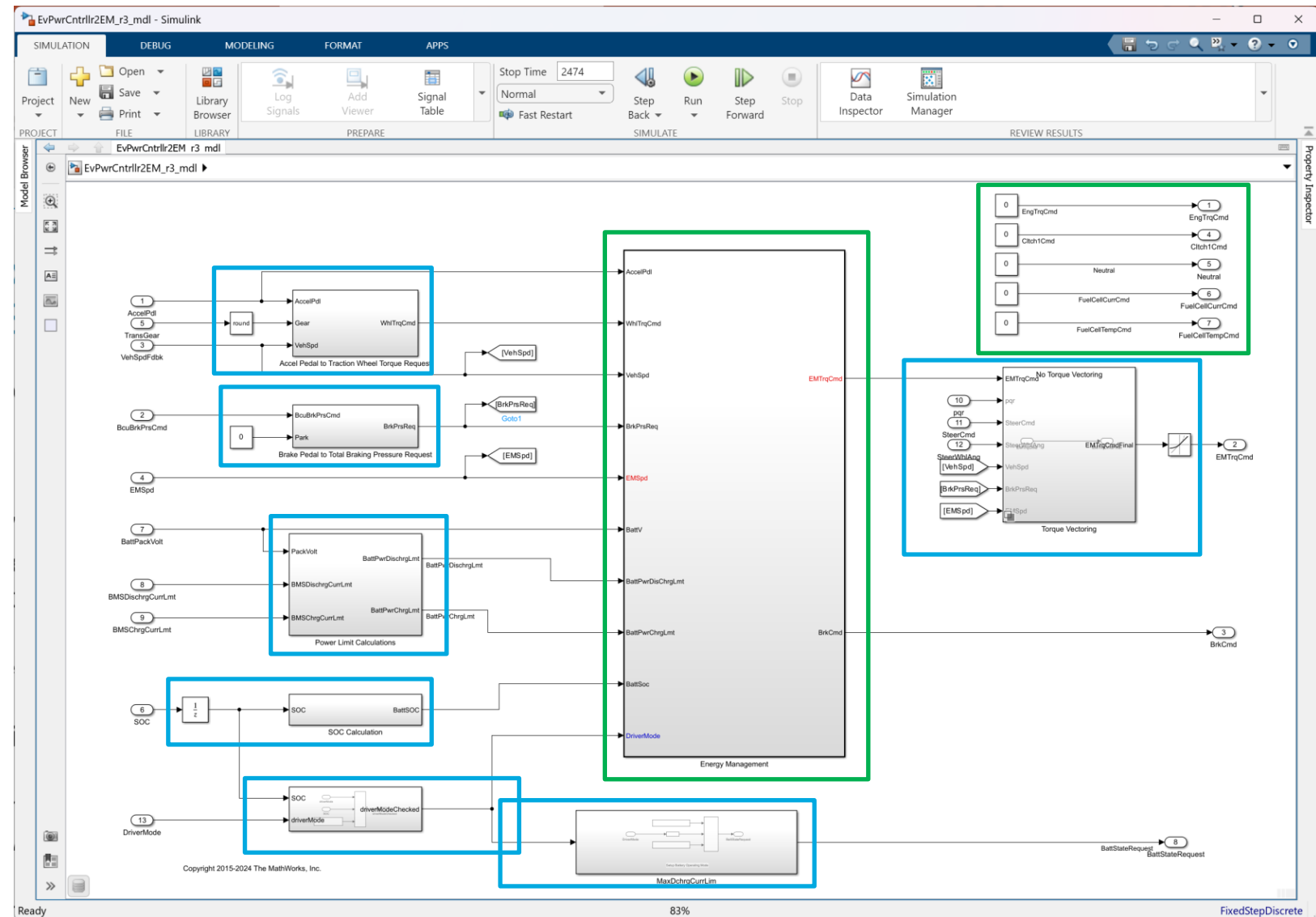
MathWorks Design Decisions

Identify and Analyze Services

For this example, we are going to refactor all of the functions that are in blue boxes into service components

In a staged approach, goal is to make a few changes, and verify with tests that no functionality was lost.

Use System Composer to refactor our design to include service components



Identify and Analyze Services

Define Services and its interfaces

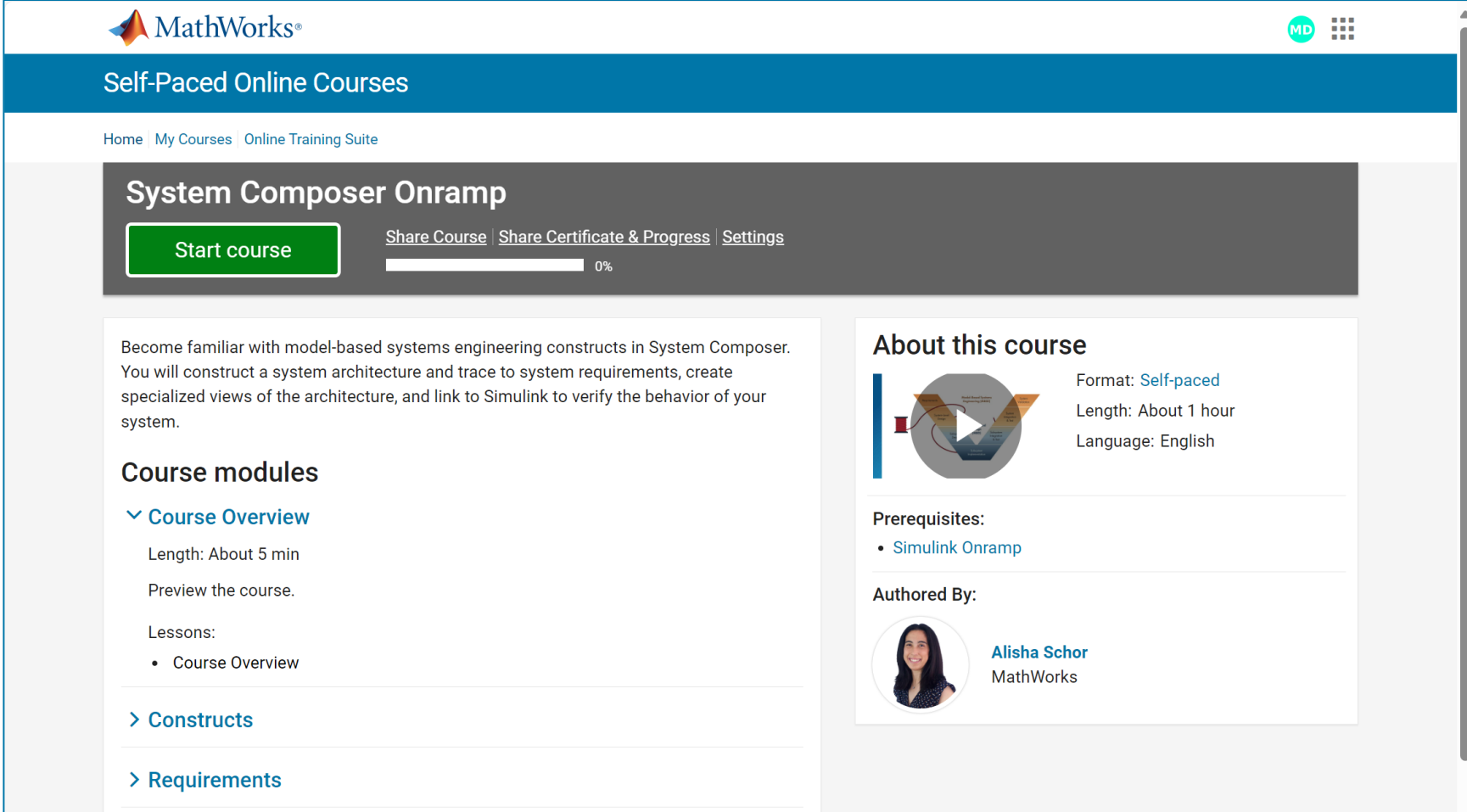
Define Service Contracts

Implement and deploy Services

New System Composer Onramp Training

Newly Release
March 2024 w/ R2024a

System Composer Onramp
is free self paced, online
training that will introduce
you to System Composer
features



The screenshot shows the MathWorks Self-Paced Online Courses page for the System Composer Onramp course. The page features a blue header with the MathWorks logo and a navigation bar with links for Home, My Courses, and Online Training Suite. The course title "System Composer Onramp" is prominently displayed, along with a green "Start course" button and a progress bar at 0%. Below the title, there are links for "Share Course", "Share Certificate & Progress", and "Settings". The main content area is divided into two columns. The left column contains a description of the course, "Course modules" (with "Course Overview" expanded to show a 5-minute length and a list of lessons), and links for "Constructs" and "Requirements". The right column contains "About this course" information, including a course icon, format (Self-paced), length (About 1 hour), language (English), prerequisites (Simulink Onramp), and the author's name (Alisha Schor, MathWorks).

MathWorks®

Self-Paced Online Courses

Home | My Courses | Online Training Suite

System Composer Onramp

[Start course](#) [Share Course](#) [Share Certificate & Progress](#) [Settings](#)

0%

Become familiar with model-based systems engineering constructs in System Composer. You will construct a system architecture and trace to system requirements, create specialized views of the architecture, and link to Simulink to verify the behavior of your system.

Course modules

▼ [Course Overview](#)

Length: About 5 min

Preview the course.


Lessons:

- [Course Overview](#)

> [Constructs](#)

> [Requirements](#)

About this course



Format: [Self-paced](#)


Length: About 1 hour

Language: English

Prerequisites:

- [Simulink Onramp](#)

Authored By:

 **Alisha Schor**
MathWorks

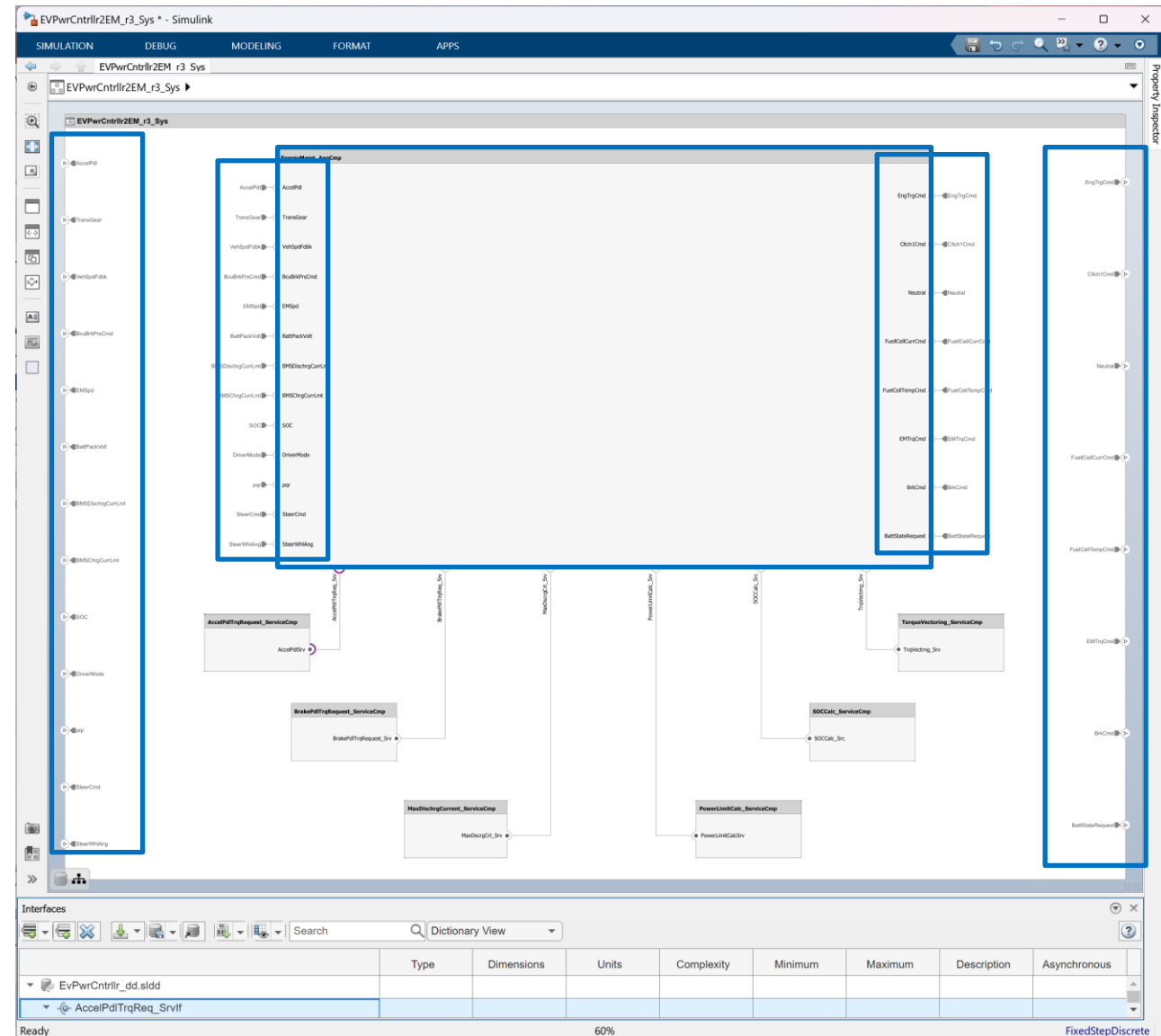
[System Composer Onramp](#)

Partition Algorithm Model Into Smaller parts with Service Interface

Define Services and Interfaces

In System Composer:

- 1) Created software architecture model
- 2) Create a software component box for our main application
- 3) Define all I/O in the main application and connect to Interface boundary of composition

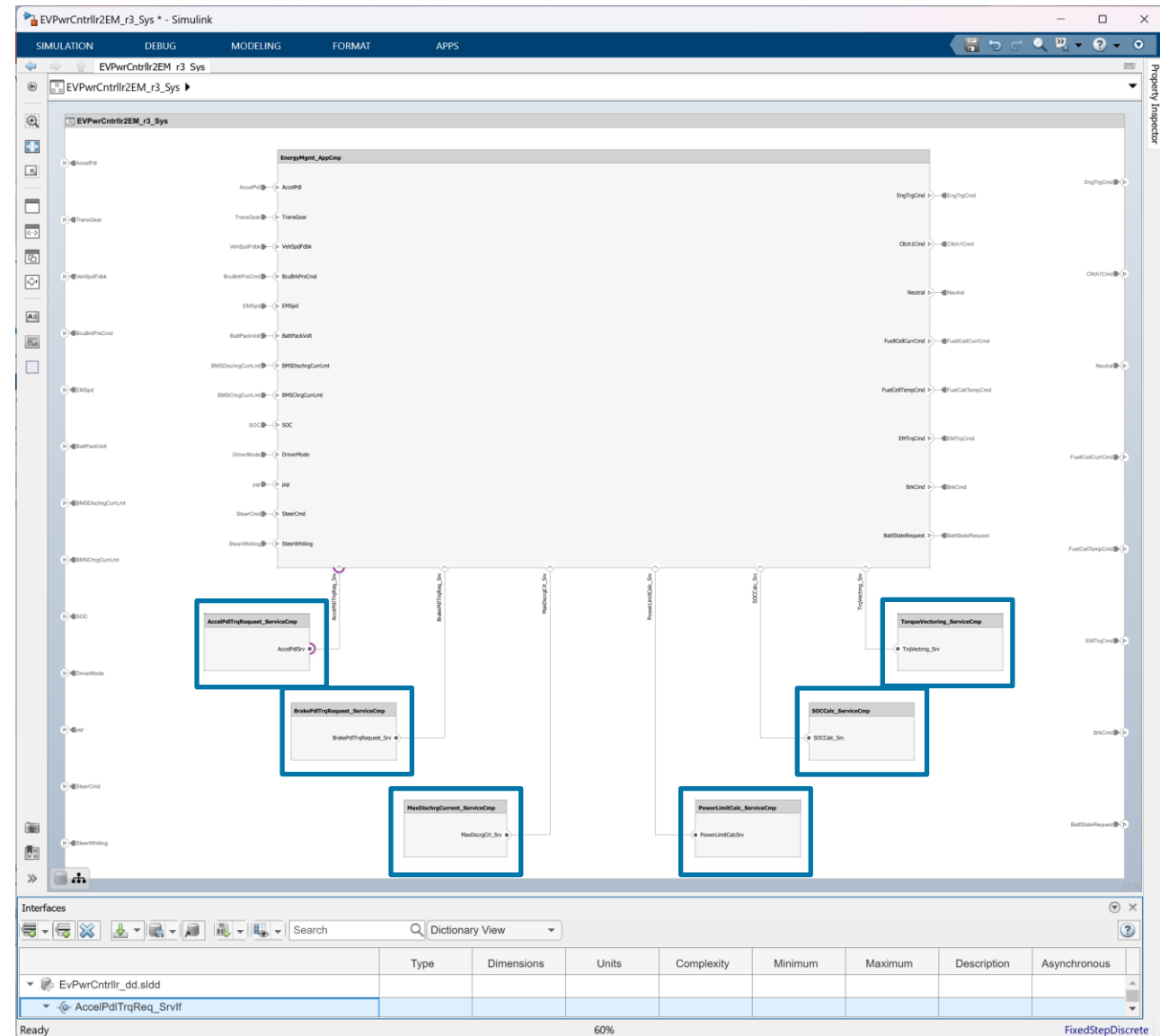


Partition Algorithm Model Into Smaller parts with Service Interface

Define Services and Interfaces

In System Composer:

- 1) Created software architecture model
- 2) Create a software component box for our main application
- 3) Define all I/O in the main application and connect to Interface boundary of composition
- 4) Create a software component boxes for all new service components



Identify and Analyze Services

Define Services and its interfaces

Define Service Contracts

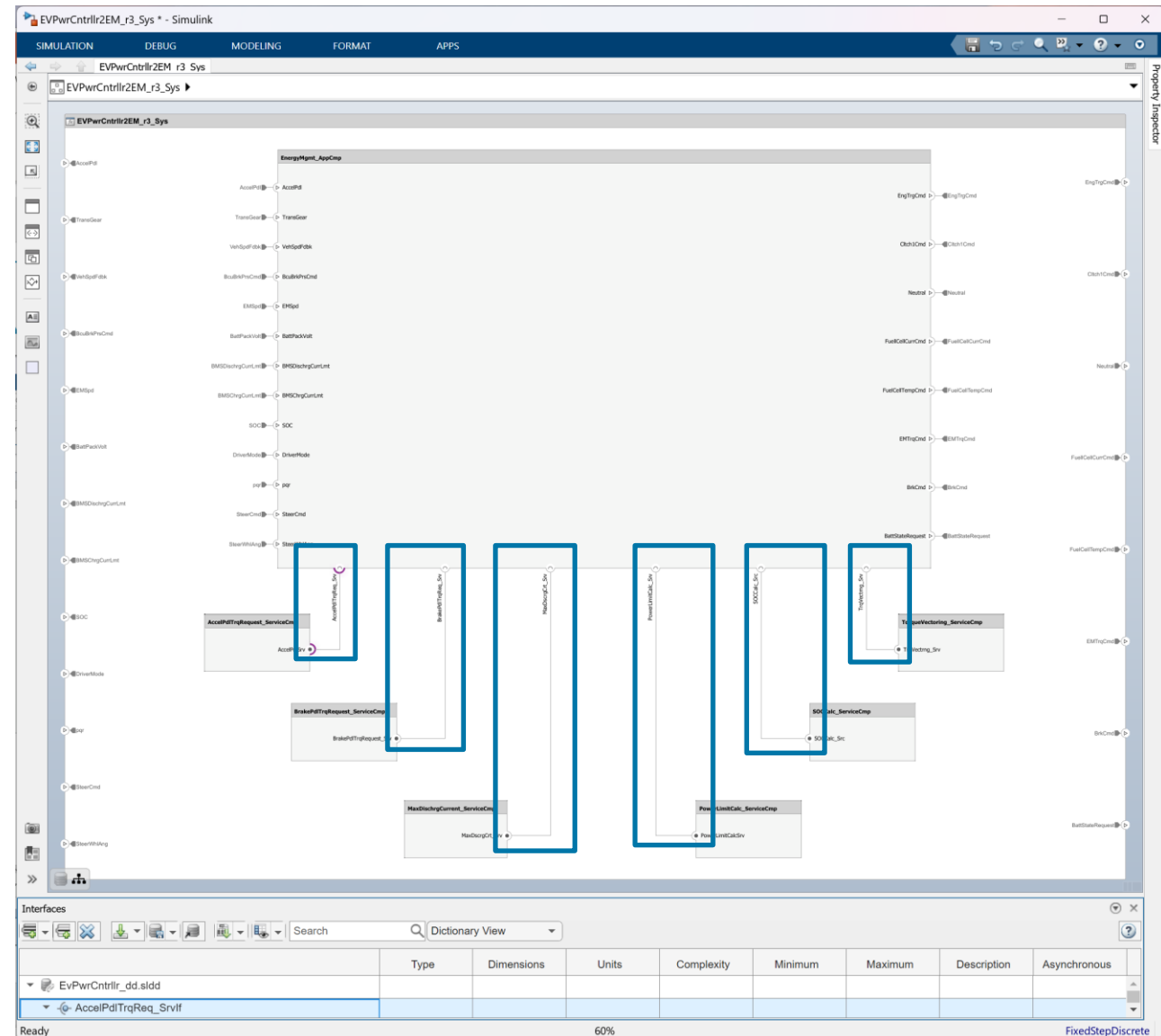
Implement and deploy Services

Partition Algorithm Model Into Smaller parts with Service Interface

Define Services and Interfaces

In System Composer:

- 1) Created software architecture model
- 2) Create a software component box for our main application
- 3) Define all I/O in the main application and connect to Interface boundary of composition
- 4) Create a software component boxes for all new service components
- 5) Connect all service components to main application with client server connectors



Identify and Analyze Services

Define Services and its interfaces

Define Service Contracts

Implement and deploy Services

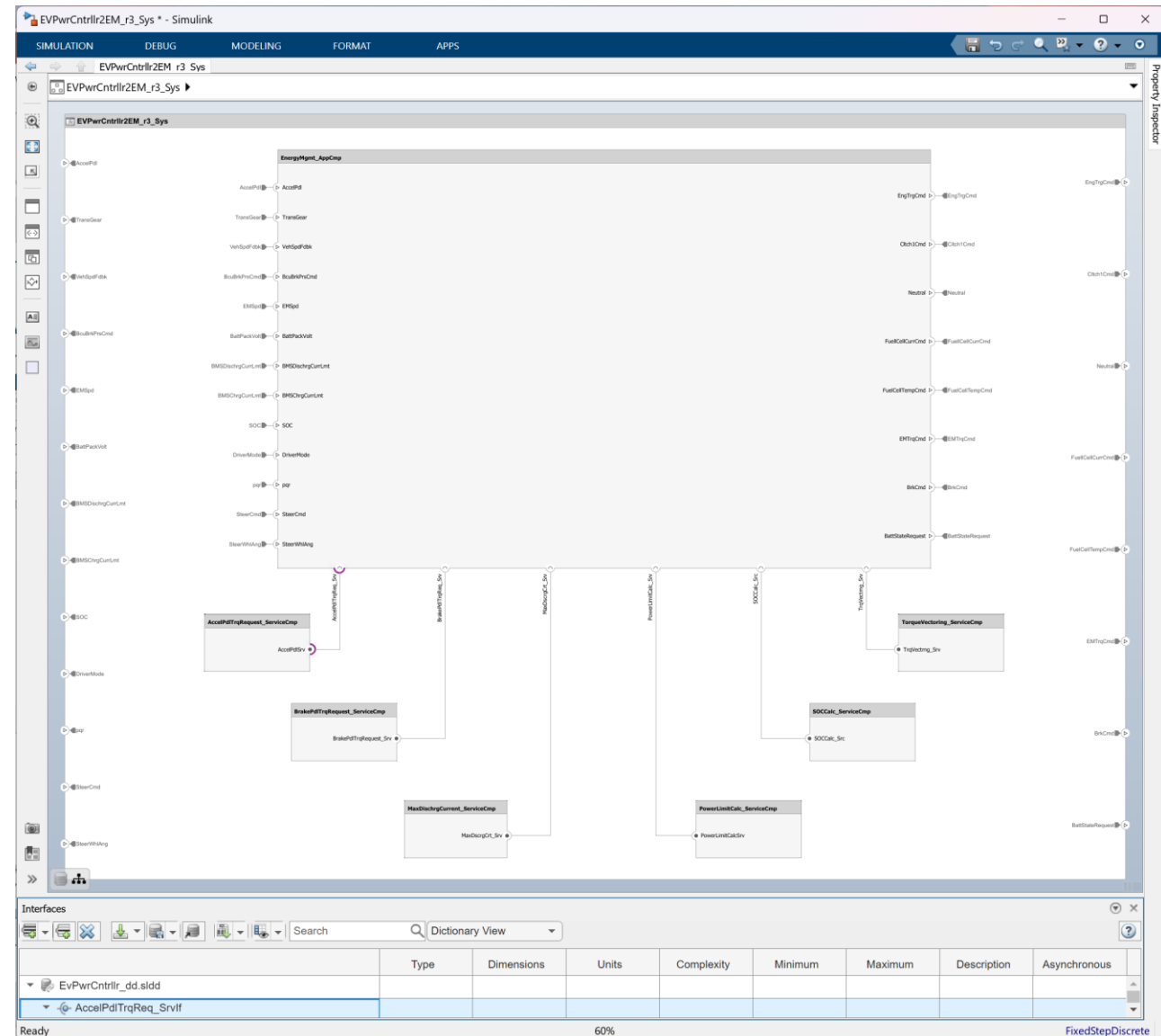
Partition Algorithm Model Into Smaller parts with Service Interface

Define Services and Interfaces

In System Composer:

- 1) Created software architecture model
- 2) Create a software component box for our main application
- 3) Define all I/O in the main application and connect to Interface boundary of composition
- 4) Create a software component boxes for all new service components
- 5) Connect all service components to main application with client server connectors

Now we have our Architecture model drawn
Next step is to create Client Server Interfaces



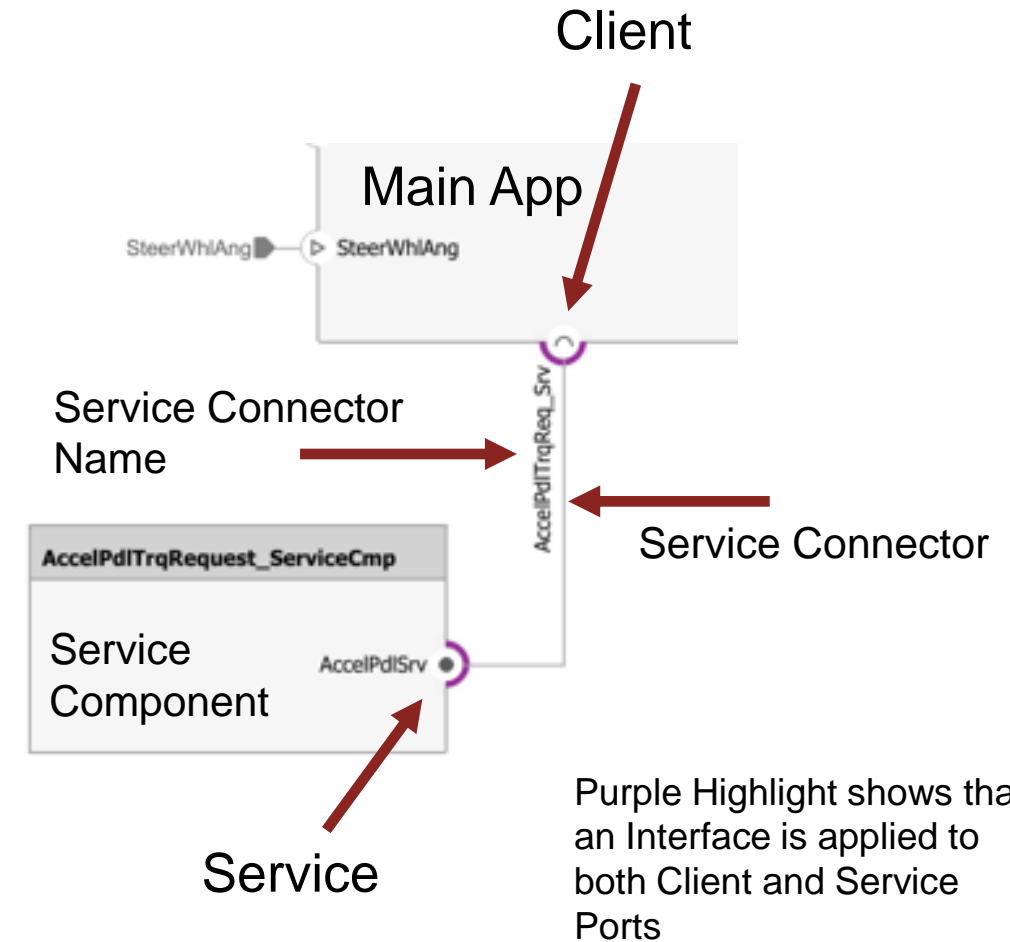
Define Client Server Interfaces

The screenshot shows a Simulink model for 'EVPwrCtrlr2EM_r3_Sys'. A blue box highlights the 'AccelPdITrqRequest_ServiceCmp' component, which contains an 'AccelPdITrqReq_Srv' port. Other components like 'SteerWhiAng' and 'BrakePdITrqRequest_ServiceCmp' are also visible. Below the model is an 'Interfaces' table.

	Type	Dimensions	Units	Complexity	Minimum	Maximum	Description	Asynchronous
EvPwrCtrlr_dd.slidd								
AccelPdITrqReq_SrvIf								
WhiTrqCmd = AccelPdITrqFcn(AccelPdI, Gear, VehSpd)								<input type="checkbox"/>
AccelPdI	double	1		real	[]	[]		
Gear	double	1		real	[]	[]		
VehSpd	double	1		real	[]	[]		
WhiTrqCmd	double	1		real	[]	[]		
BrakePdITrqReq_SrvIf								
BrkPrsReq = BrkPdITrqFcn(BcuBrkPrsCmd, Park)								<input type="checkbox"/>

Define Services and Interfaces

A Service Component is connected to the Main App through a Service Connector between the Client Port and a Service Port



Define Client Server Interfaces

Define Services and Interfaces

Ability to define the full Service-Interface in terms of the function prototypes and arguments

This allows for the Service Interface to be used to create the model constructs for both the main app component and the Service Component

Define the Client / Service Interface for each of the connected Service Components

The screenshot shows a Simulink model with several service connectors. A callout box labeled "Service Connector Name" points to a connector labeled "AccelPdITrqReq_Srv". Another callout box labeled "Service Interface Name" points to the "AccelPdITrqReq_SrvIf" entry in the "Interfaces" table.

The "Interfaces" table is shown in two views: a compact view and a detailed "Dictionary View".

Interface Name	Type	Dimensions	Unit
EvPwrCtrlr_dd.sldd			
- AccelPdITrqReq_SrvIf			
- WhlTrqCmd = AcclPdIToTrqFcn(AccelPdI,Gear,VehSpd)			
AccelPdI	double	1	
Gear	double	1	
VehSpd	double	1	
WhlTrqCmd	double	1	
- BrakePdITrqReq_SrvIf			
- BrkPrsReq = BrkPdIToTrqFcn(BcuBrkPrsCmd, Park)			

Function based Proto-Type shows Function Name and Input / Output Arguments

Define Inputs and Outputs to function in terms of datatypes, dimensions, units, min, max etc;

Create Simulink Behavior Model for Service Component

Creating the Simulink Behavior (empty skeleton model that will contain your algorithm)

Right click on Component boxes to create the Simulink Behavior

This action creates a skeleton or shell model

Also, this will create blocks that represent the Client / Service interfaces

The screenshot displays the Simulink environment with a context menu open over a component box. The menu options are:

- Explore
- Open In New Tab
- Open In New Window
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Create Architecture...
- Create Simulink Behavior...** (highlighted)
- Link to Model...
- Add Variant Choice
- Apply Stereotype
- Change Stereotype
- Allocations
- Requirements
- Test Harness
- Create Spotlight From Component
- Format
- Properties...
- Help

The background shows a Simulink model with several service component blocks connected by lines. The blocks include:

- AccelPdTrqRequest_ServiceCmp
- BrakePdTrqRequest_ServiceCmp
- MaxDischrgCurrent_ServiceCmp
- PowerLimitCalc_ServiceCmp
- SOCCalc_Ser
- SOCCalc_Src

At the bottom, a table shows the properties of the selected component:

	Type	Dimensions	Units	Complexity	Minimum	Maximum	Description
BrkPrsReq = BrkPdItoTrqFcn(BcuBrkPrsCmd, Park)							
BcuBrkPrsCmd	double	4		real	[]	[]	
Park	double	1		real	[]	[]	
BrkPrsReq	double	4		real	[]	[]	
MaxDischrgCrt_SrvIf							

Create Simulink Behavior Model for Service Component

The image shows a Simulink workspace with two windows. The main window displays a subsystem block named 'BrakePdITrqRequest_ServiceCmp'. Inside this block, a function block is defined with the following code:

```
BrkPrsReq = BrkPdITrqFcn(BcuBrkPrsCmd, Park)
```

The function block has two input ports and one output port. The output port is connected to a service connector named 'BrakePdITrqRequest_Srv . BrkPdITrqFcn'. The service connector name is highlighted with a callout. The function name 'BrkPdITrqFcn' is also highlighted with a callout. The function block itself is highlighted with a callout. The new block being added is highlighted with a callout.

Callouts in the image point to the following elements:

- New Block: Function Port**: Points to the function block being added to the subsystem.
- Service Connector Name**: Points to the service connector 'BrakePdITrqRequest_Srv . BrkPdITrqFcn'.
- Simulink Function**: Points to the function definition 'BrkPrsReq = BrkPdITrqFcn(BcuBrkPrsCmd, Park)'.
- Simulink Function Name**: Points to the function name 'BrkPdITrqFcn'.

The 'Interfaces' window at the bottom left shows the following table:

Interface	Type
BrkPrsReq = BrkPdITrqFcn(BcuBrkPrsCmd, Park)	
BcuBrkPrsCmd	double
Park	double
BrkPrsReq	double

Create Simulink Behavior Model for Service Component

The image displays three Simulink windows illustrating the creation of a behavior model for a service component.

Top Window: BrakePdTrqRequest_ServiceCmp
 This window shows the subsystem block for the service component. A callout points to the function signature: `BrakePdTrqRequest_Srv . BrkPdToTrqFcn`.

Middle Window: BrakePdToTrqFcn
 This window shows the function block with the signature: `BrkPrsReq = BrkPdToTrqFcn(BcuBrkPrsCmd, Park)`. The block is currently empty.

Bottom Window: BrakePdToTrqFcn (Detailed)
 This window shows the internal structure of the function block. It contains an empty function block labeled `f()` and the name `BrkPdToTrqFcn`. The input signals are `BcuBrkPrsCmd` and `Park`, and the output signal is `BrkPrsReq`.

Interfaces Table:

Interface	Type	Direction	Value	Real	Complex	Integer	Fixed-Point	String	Enumeration	Port	Ready	Progress
BrkPrsReq = BrkPdToTrqFcn(BcuBrkPrsCmd, Park)											Ready	337%
BcuBrkPrsCmd	double											
Park	double	1		real								
BrkPrsReq	double	1		real								

Empty Simulink Function

Create Simulink Behavior Model for Service Component

Copy/Paste Original Function Content into our Simulink Function Here

$$\text{BrkPrsReq} = \text{BrkPdITrqFcn}(\text{BcuBrkPrsCmd}, \text{Park})$$

BrkPdITrqFcn

Front Bias
Rear Bias

PlntBrkFmtBias
PlntBrkRearBias
Rear Bias

BcuBrkPrsCmd
Park

BrkPdITrqFcn

BrkPrsReq

Interface	Type	Value	Unit	Real	Complex	Integer	Enumeration
BrkPrsReq = BrkPdITrqFcn(BcuBrkPrsCmd, Park)							
BcuBrkPrsCmd	double						
Park	double	1		real			
BrkPrsReq	double	1		real			

Create Simulink Behavior Model for Main Application Component

EV Pwr Cntrlr2EM_r3 Sys - Simulink

EXPLORE
OPEN IN NEW TAB
OPEN IN NEW WINDOW
CUT Ctrl+X
COPY Ctrl+C
PASTE Ctrl+V
CREATE ARCHITECTURE...
CREATE SIMULINK BEHAVIOR...
LINK TO MODEL...
ADD VARIANT CHOICE
APPLY STEREOYPE
CHANGE STEREOYPE
ALLOCATIONS
REQUIREMENTS
TEST HARNESS
CREATE SPOTLIGHT FROM COMPONENT
FORMAT
PROPERTIES...
HELP

Interfaces

	Type	Dimensions	Units	Complexity	Minimum	Maximum	Description	Asynchronous
EV Pwr Cntrlr_dd.sldd								
AccelPdTrqReq_SrvIf								

Ready 60% FixedStepDiscrete

Created skeleton model

EV Pwr Cntrlr2EM_r3 Sys - Simulink

Interfaces

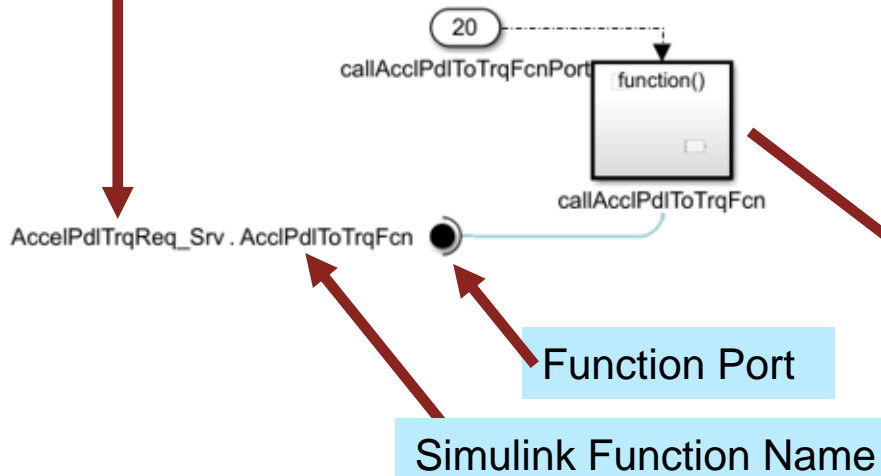
	Type	Dimensions	Units	Complexity	Minimum	Maximum	Description	Asynchronous
EV Pwr Cntrlr_dd.sldd								
AccelPdTrqReq_SrvIf								

Ready 60% FixedStepDiscrete

Create Simulink Behavior Model for Main Application Component

Service Connector Name

Triggered Subsystem



Inputs and Outputs

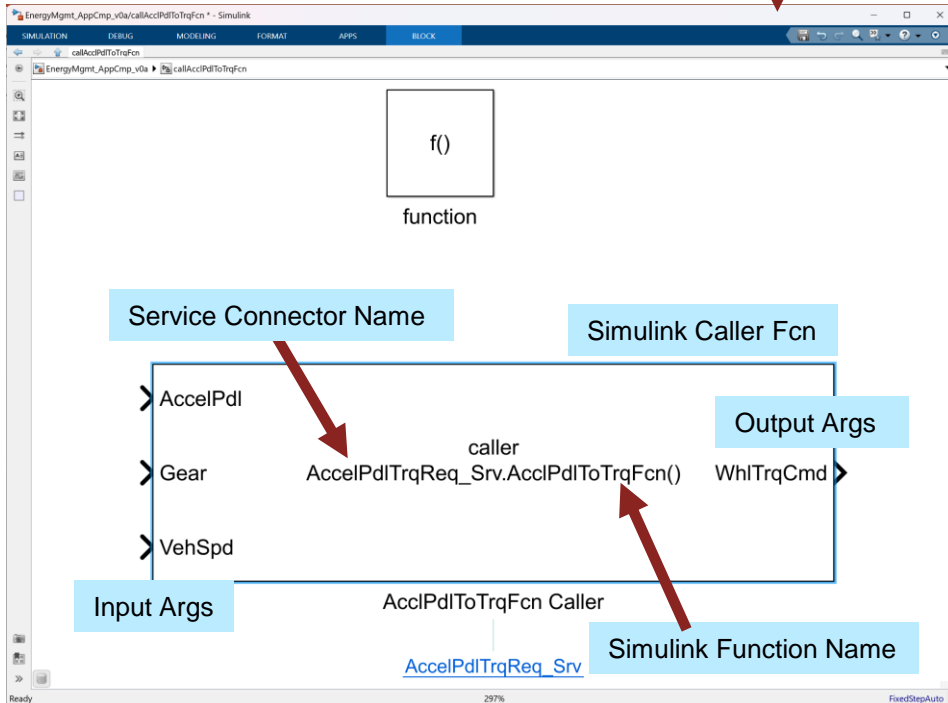
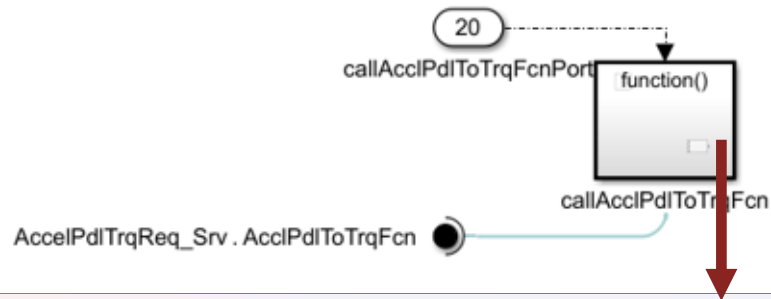
- AccelPd
- TransGear
- VehSpdFdbk
- BouBrkPrsCmd
- EMSpd
- BattPackVolt
- BMSDischrgCurrLmt
- BMSChrgCurrLmt
- SOC
- DriverMode
- pqr
- SteerCmd
- SteerWhiAng

Triggered Subsystem w/ Function Caller inside

- callAcclPdToTrqFcnPort (20) → function() → callAcclPdToTrqFcn → AccelPdTrqReq_Srv . AcclPdToTrqFcn
- callBrkPdToTrqFcnPort (21) → function() → callBrkPdToTrqFcn → BrakePdTrqReq_Srv . BrkPdToTrqFcn
- callPwrLimitFcnPort (22) → function() → callPwrLimitFcn → PowerLimitCalc_Srv . PwrLimitFcn
- callSOCCalcFcnPort (23) → function() → callSOCCalcFcn → SOCCalc_Src . SOCCalcFcn
- callTrqVectrngFcnPort (24) → function() → callTrqVectrngFcn → TrqVectrng_Srv . TrqVectrngFcn
- callMaxDchrgCurrLimFcnPort (25) → function() → callMaxDchrgCurrLimFcn → MaxDchrgCrt_Srv . MaxDchrgCurrLimFcn

Create Simulink Behavior Model for Main Application Component

Triggered Subsystem

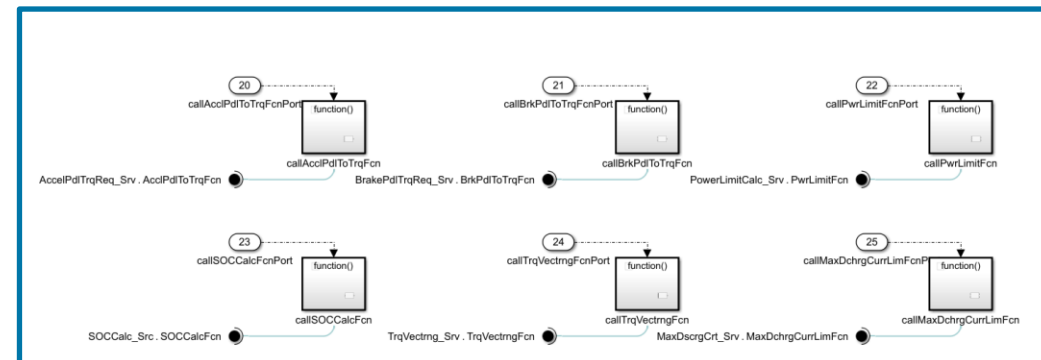


Inputs and Outputs

- AccelPdi
- TransGear
- VehSpdFdbk
- BcuBrkPrsCmd
- EMSpd
- BattPackVolt
- BMSDischrgCurrLmt
- BMSChrgCurrLmt
- SOC
- DriverMode
- pqr
- SteerCmd
- SteerWhlAng

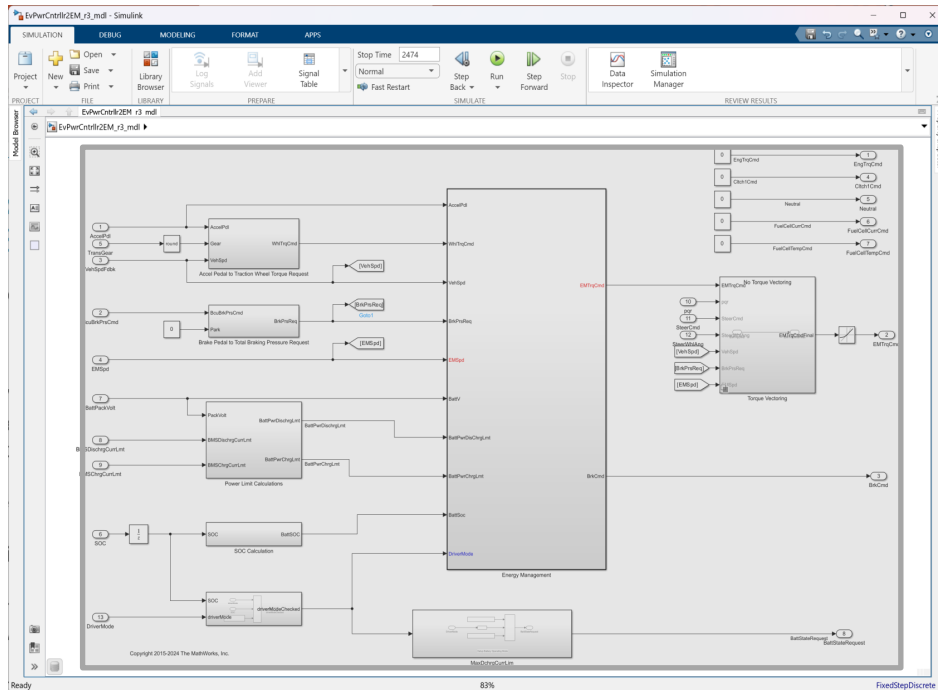
- EngTrgCmd
- Cltch1Cmd
- Neutral
- FuelCellCurrCmd
- FuelCellTempCmd
- EMTrqCmd
- BrkCmd
- BattStateRequest

Triggered Subsystem w/ Function Caller inside



Create Simulink Behavior Model for Main Application Component

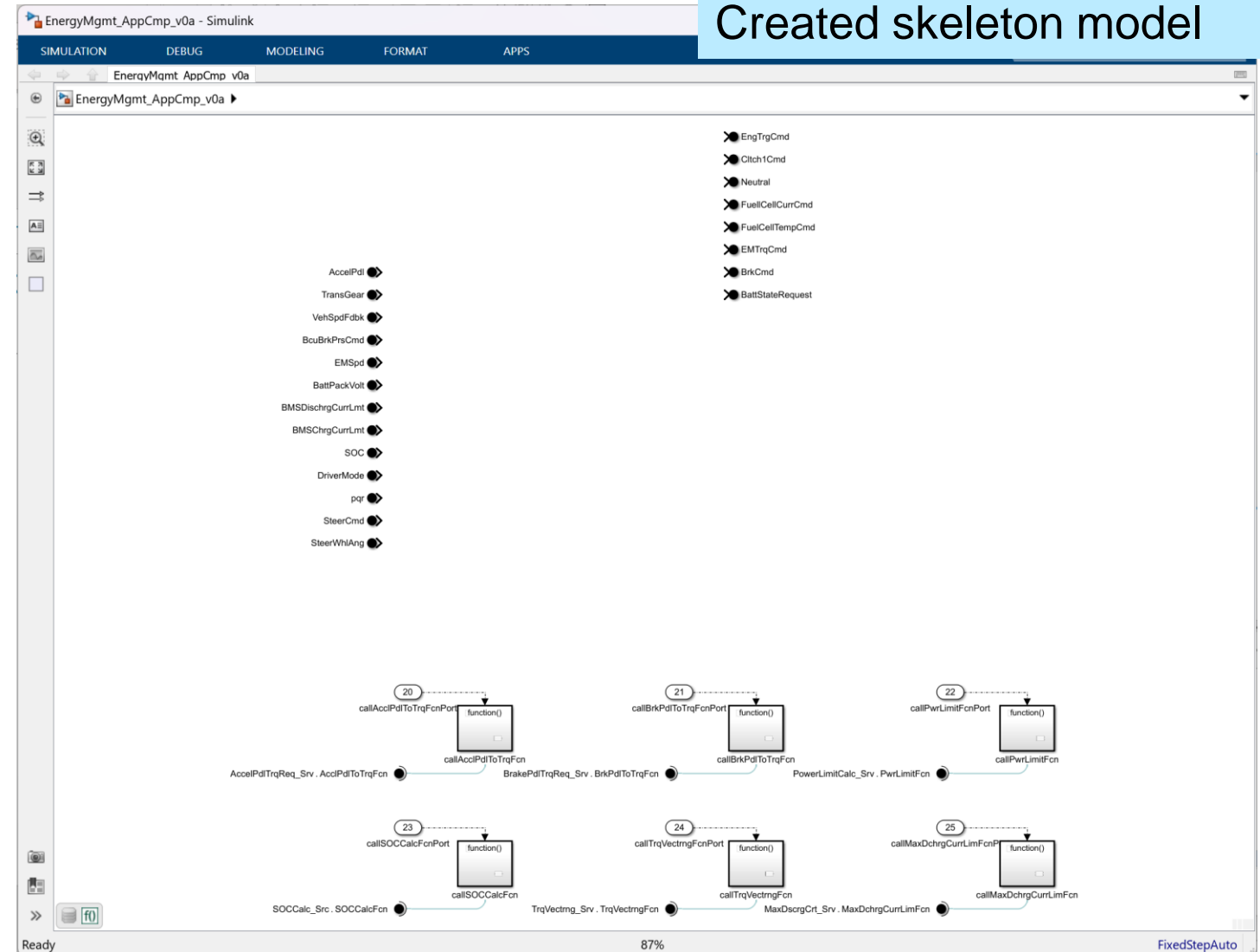
Original Model



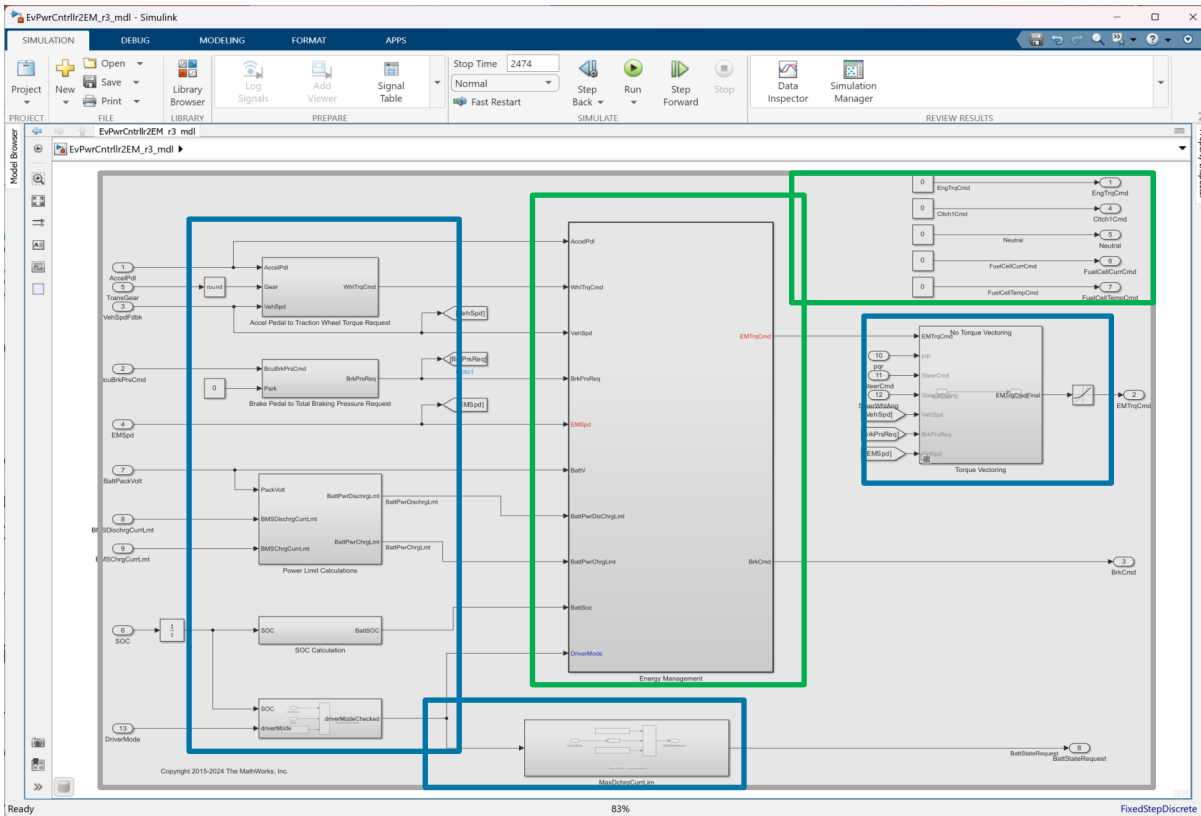
Process:

- Copy/paste parts from original model that we are keeping
- Alter parts that represent the client / caller functions

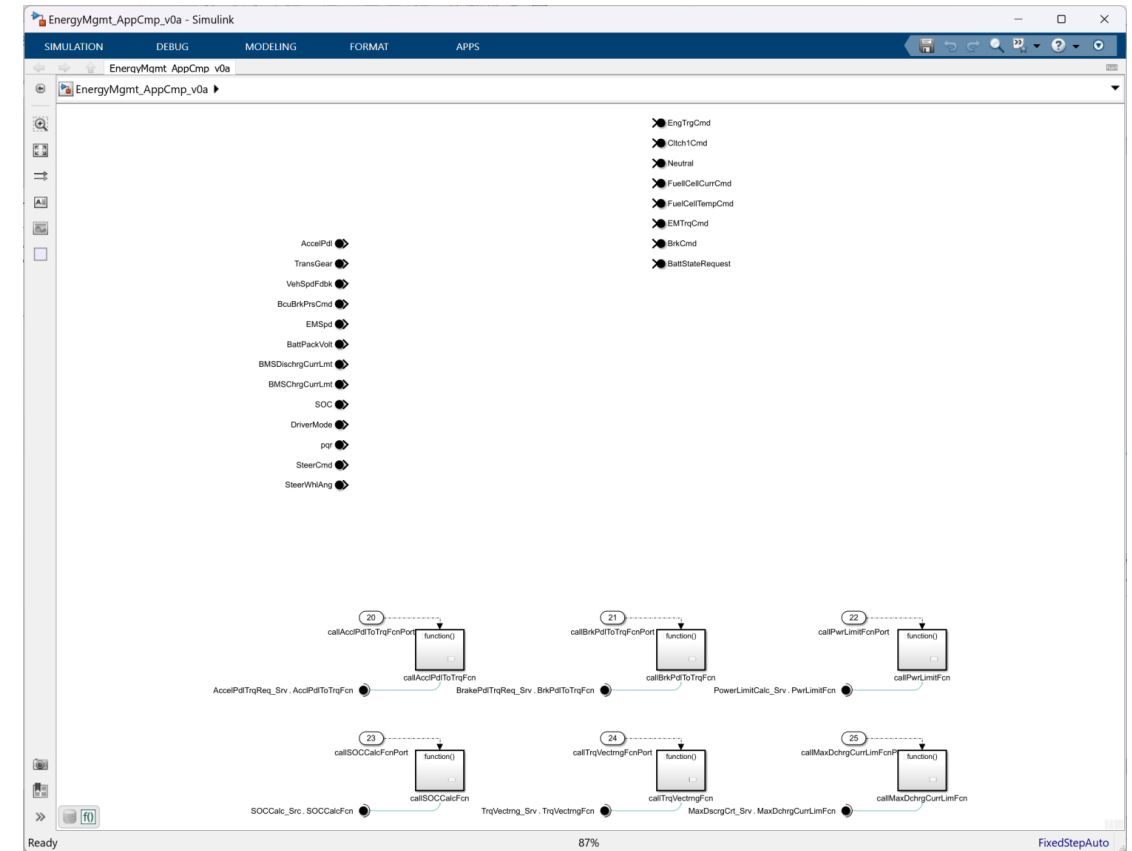
Created skeleton model



Create Simulink Behavior Model for Main Application Component



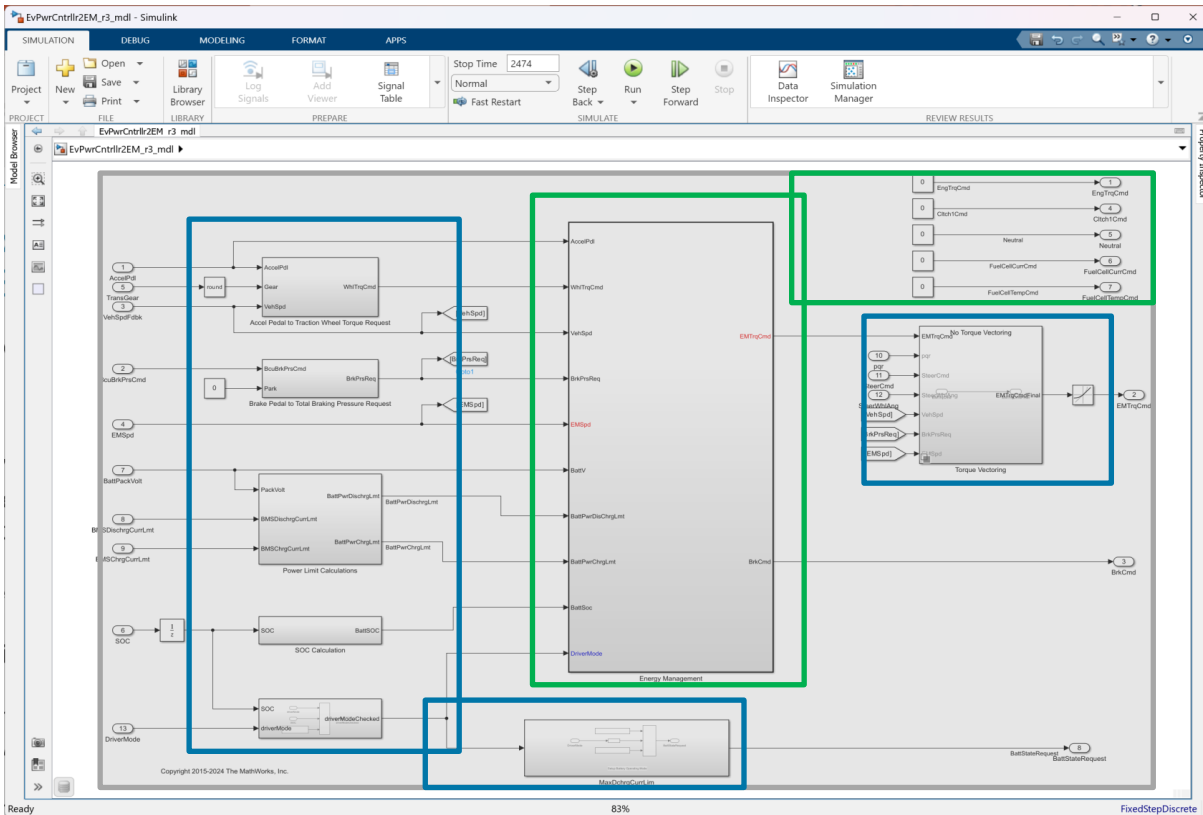
Original Model



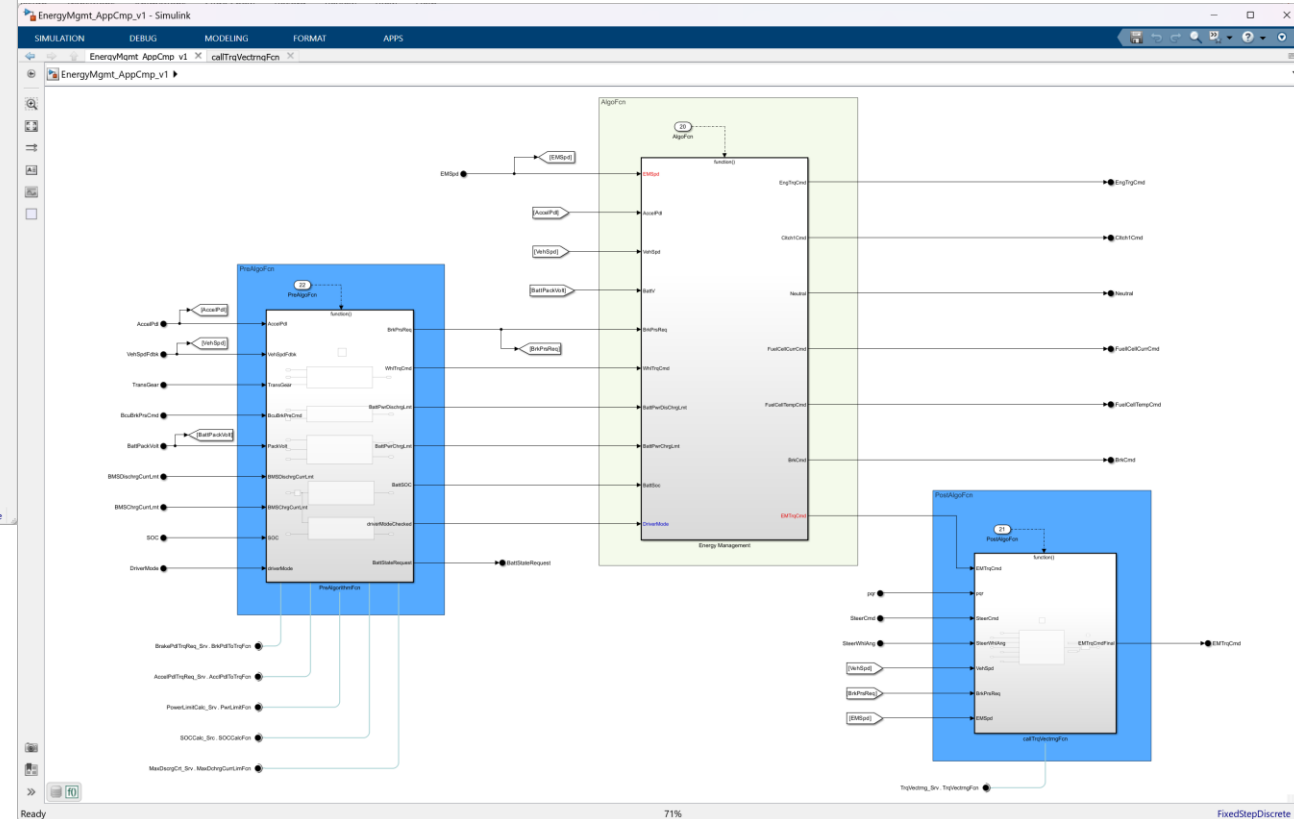
Transform this shell model to capture original intent

Create Simulink Behavior Model for Main Application Component

New Simulink behavior model that captures Application logic and Service Call outs

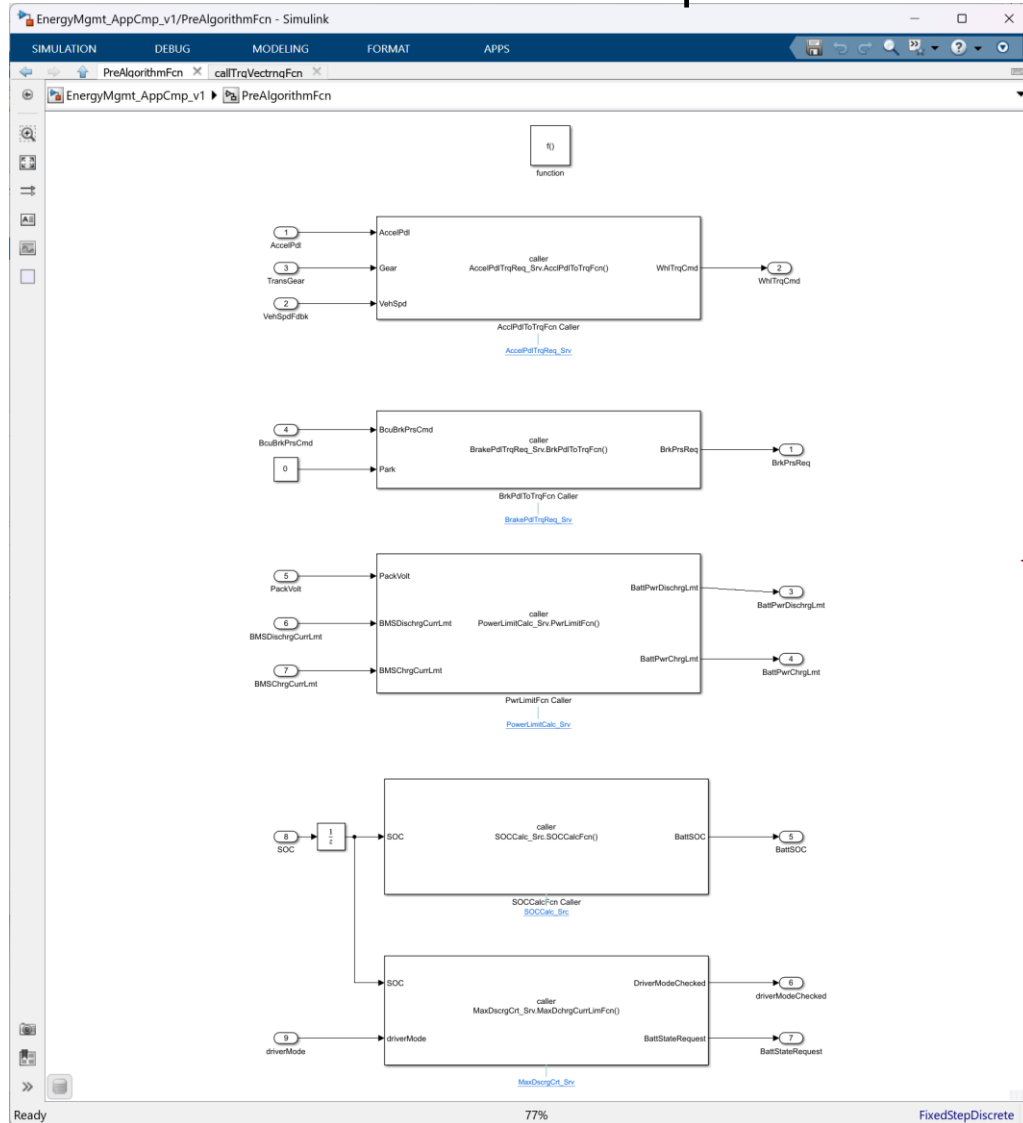


Original Model

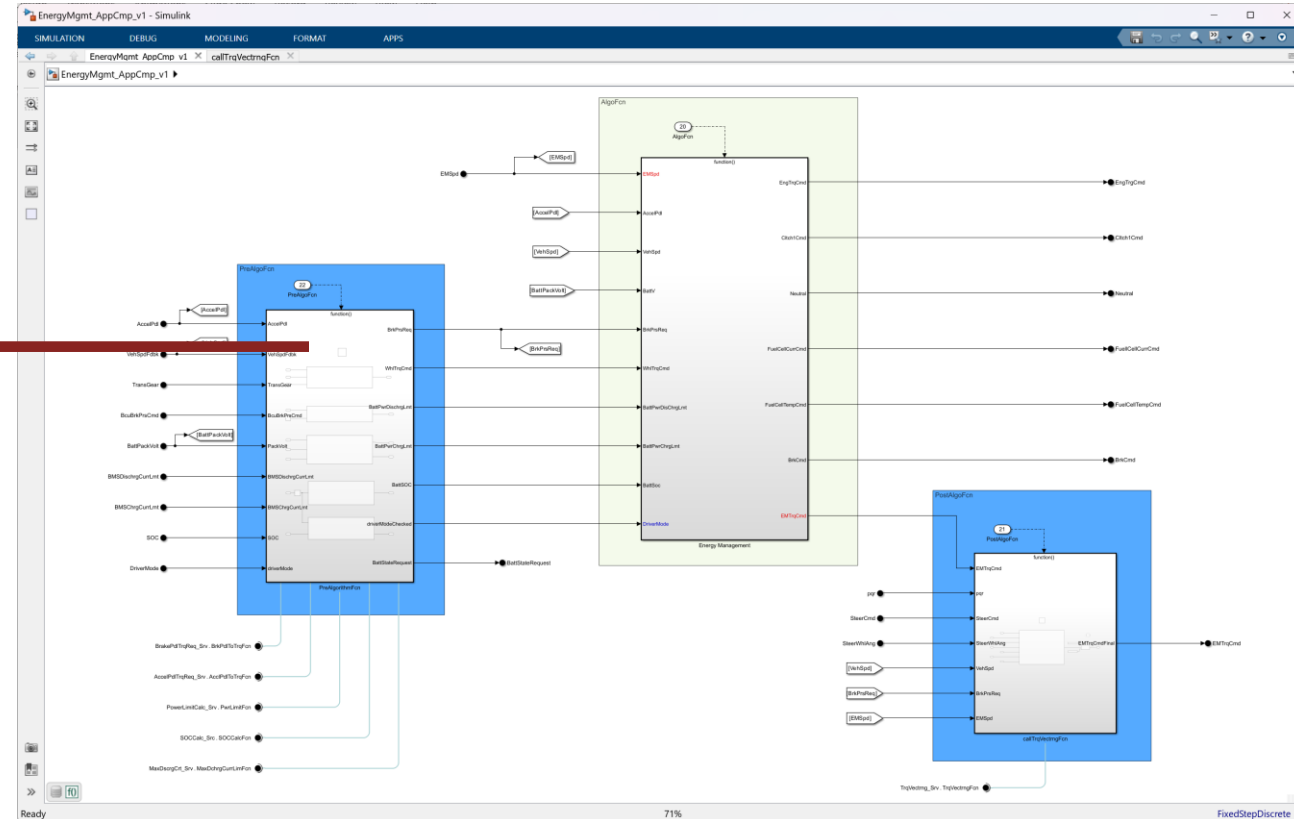


Create Simulink Behavior Model for Main Application Component

Call out to Service Components



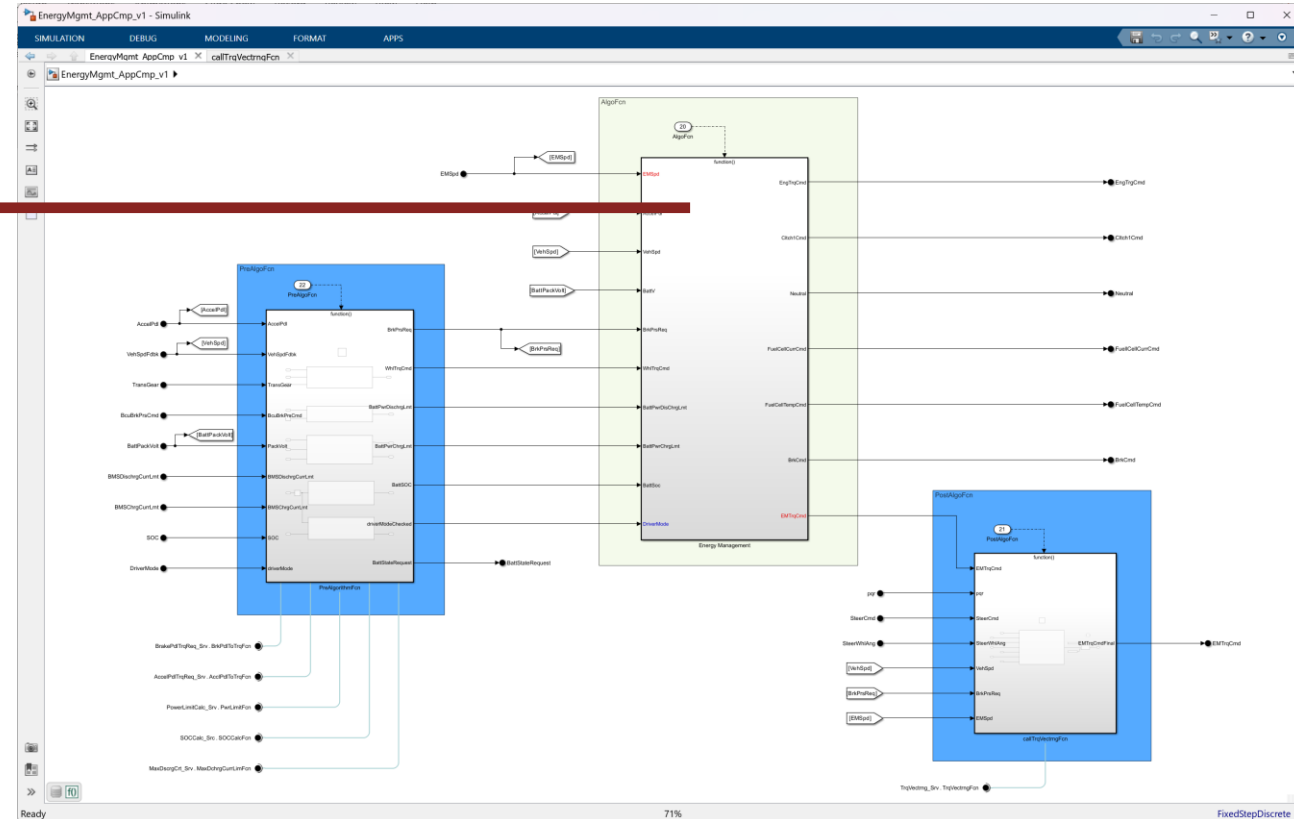
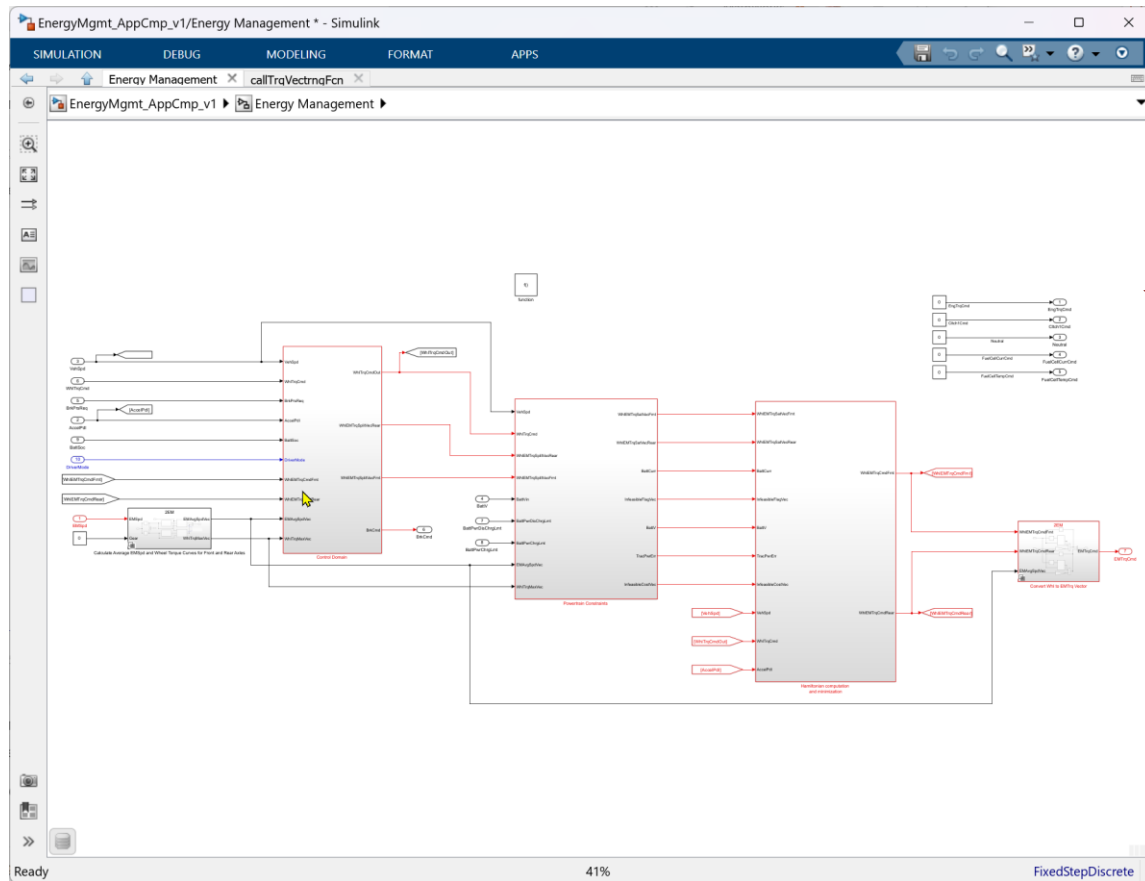
New Simulink behavior model that captures Application logic and Service Call outs



Create Simulink Behavior Model for Main Application Component

Original Logic that we left alone

New Simulink behavior model that captures Application logic and Service Call outs



End refactoring process with Baseline Tests that pass

Very Important: Apply original model tests such as to prove out that functionality remains unchanged

TESTS

+ New Open Save Cut Delete Test Spec Run Run with Stepper Stop Parallel Report Visualize Highlight in Model Export Import Model Testing Dashboard Preferences

FILE EDIT RUN RESULTS ENVIRONMENT

Test Browser Results and Artifacts

Filter results by name or tags, e.g. tags: test

NAME	STATUS
Results: 2024-Apr-12 11:13:12	2 ✓
EV2M_VCU_MiLtests	2 ✓
VCU_2EMEV_ctrl_powertrain	2 ✓
VCU_2EMEV_Harness_Bas	✓
VCU_2EMEV_Harness_High	✓

PROPERTY VALUE

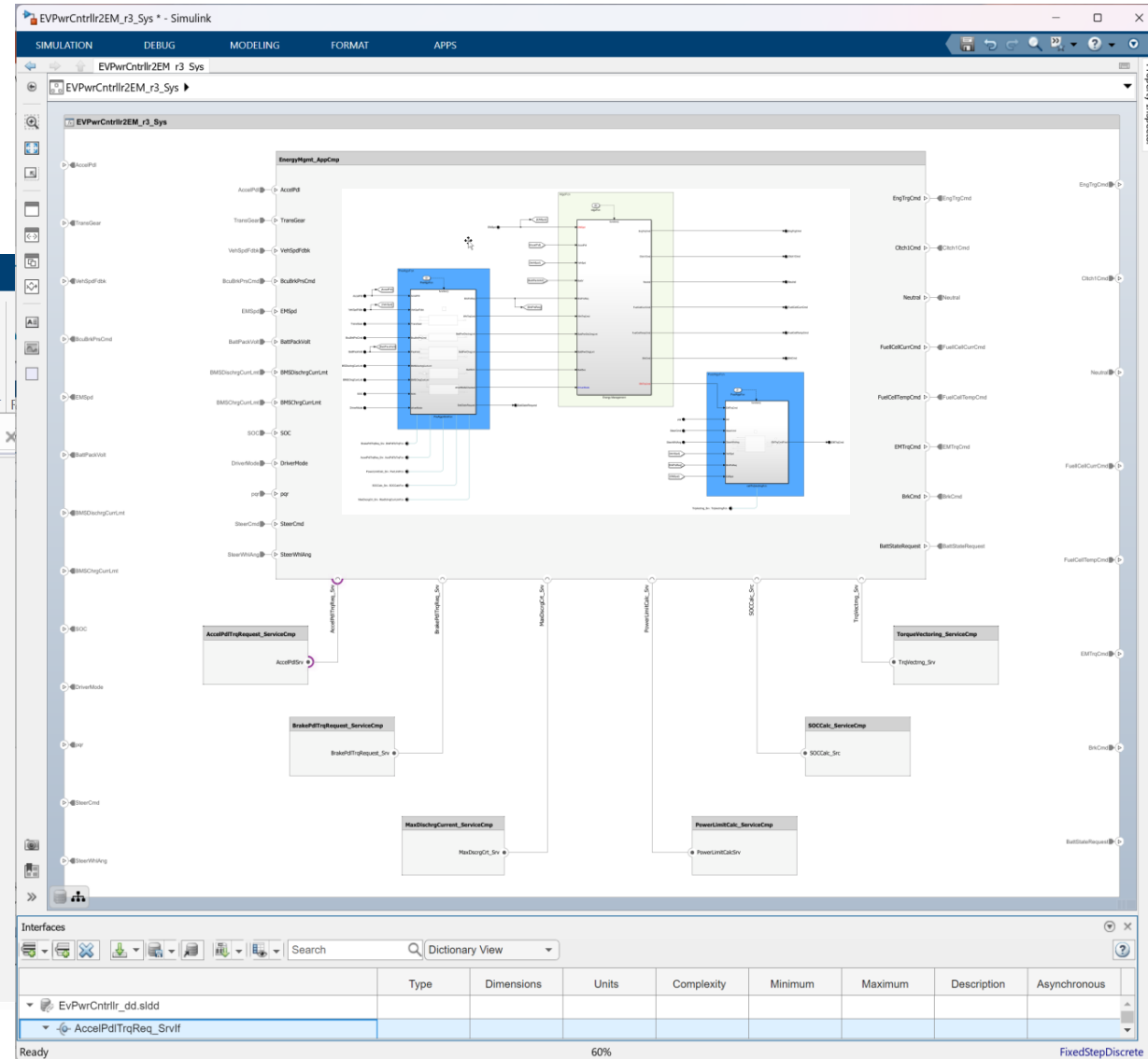
Name	VCU_2EMEV_Harness...
Status	1 ✓
Start Time	04/12/2024 11:13:13
End Time	04/12/2024 11:14:09
Type	Baseline Test

Results: 2024-Apr-12 11:13:12

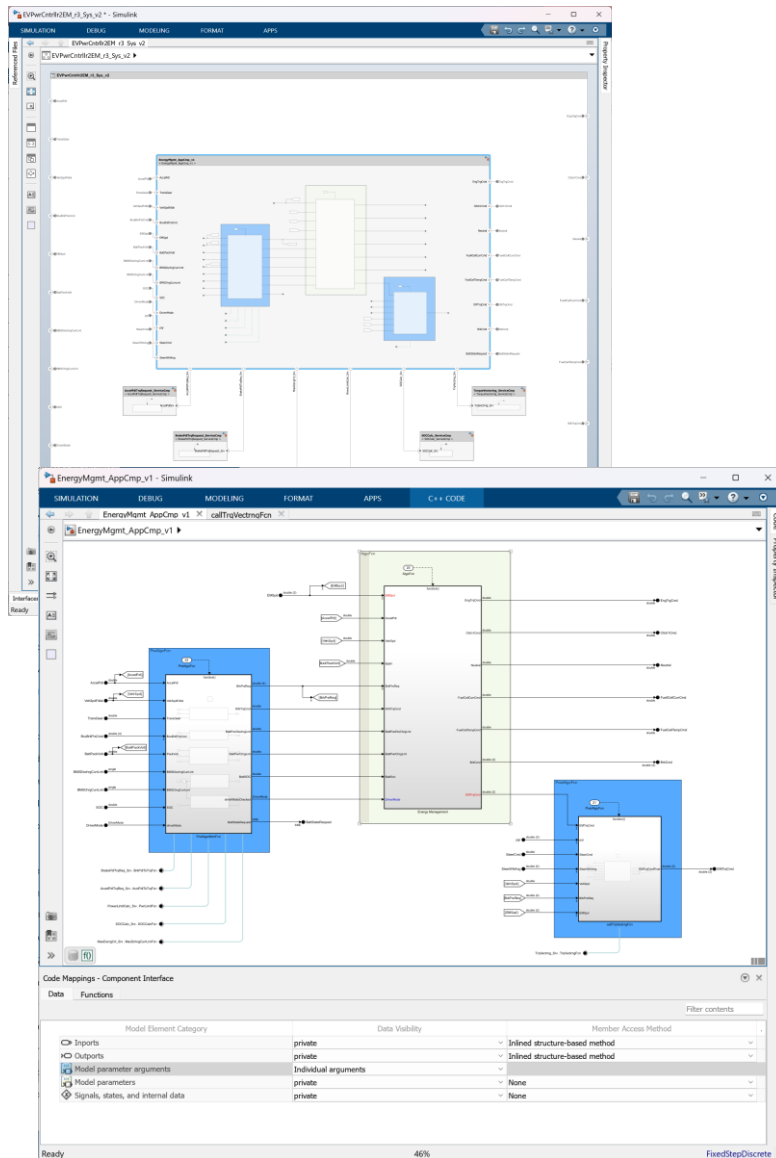
SUMMARY

Name	Results: 2024-Apr-12 11:13:12
Outcome	2 ✓
Start Time	04/12/2024 11:13:13
End Time	04/12/2024 11:15:01
Type	Result Set

Tests Passed After Model Modifications



Generate C++ code that is agnostic to any platform



Code Generation Report

Find: Match Case

Current model: EnergyMgmt_AppCmp_v1

Eliminated Blocks
Code Replacements Report
Coder Assumptions

Code

- Main file
 - ert_main.cpp
- Model files
 - EnergyMgmt_AppCmp_v1.cpp
 - EnergyMgmt_AppCmp_v1.h
 - EnergyMgmt_AppCmp_v1_private.h
 - EnergyMgmt_AppCmp_v1_types.h
- Shared files
 - AccelPdlTrqReq_SrvIfT.h
 - BrakePdlTrqReq_SrvIfT.h
 - PowerLimitCalc_SrvIfT.h
 - SOCCalc_SrcIfT.h
 - TrqVectrng_SrvIfT.h
 - binsearch_u32d_prevIdx.cpp
 - binsearch_u32d_prevIdx.h
 - const_params.cpp
 - intrap2d_la.cpp
 - intrap2d_la.h
 - look1_binlca.cpp
 - look1_binlca.h
 - look1_pbinlca.cpp
 - look1_pbinlca.h

EnergyMgmt_AppCmp_v1.cpp

```

1362
1363 // End of Outputs for RootInportFunctionCallGenerator generated from: '<Root>/PostAlgoFcn'
1364 }
1365
1366 // Model step function for TID3
1367 void EnergyMgmt_AppCmp_v1::PreAlgoFcn() // Explicit Task: PreAlgoFcn
1368 {
1369     real_T rtb_UnitDelay_p;
1370
1371     // RootInportFunctionCallGenerator generated from: '<Root>/PreAlgoFcn' incorporates:
1372     // SubSystem: '<Root>/PreAlgorithmFcn'
1373
1374     // FunctionCaller: '<S2>/BrkPdlToTrqFcn Caller' incorporates:
1375     // Constant: '<S2>/Constant'
1376     // Inport generated from: '<Root>/Bus Element In4'
1377
1378     BrakePdlTrqReq_Srv.BrkPdlToTrqFcn(EnergyMgmt_AppCmp_v1_U.BcuBrkPrsCmd, 0.0,
1379     EnergyMgmt_AppCmp_v1_DW.BrkPdlToTrqFcnCaller);
1380
1381     // FunctionCaller: '<S2>/AcclPdlToTrqFcn Caller' incorporates:
1382     // Inport generated from: '<Root>/Bus Element In1'
1383     // Inport generated from: '<Root>/Bus Element In2'
1384     // Inport generated from: '<Root>/Bus Element In3'
1385
1386     AccelPdlTrqReq_Srv.AcclPdlToTrqFcn(EnergyMgmt_AppCmp_v1_U.AccelPdl,
1387     EnergyMgmt_AppCmp_v1_U.TransGear, EnergyMgmt_AppCmp_v1_U.VehSpdFdbk,
1388     &EnergyMgmt_AppCmp_v1_DW.AcclPdlToTrqFcnCaller);
1389
1390     // FunctionCaller: '<S2>/PwrLimitFcn Caller' incorporates:
1391     // Inport generated from: '<Root>/Bus Element In8'
1392     // Inport generated from: '<Root>/Bus Element In7'
1393     // Inport generated from: '<Root>/Bus Element In6'
  
```

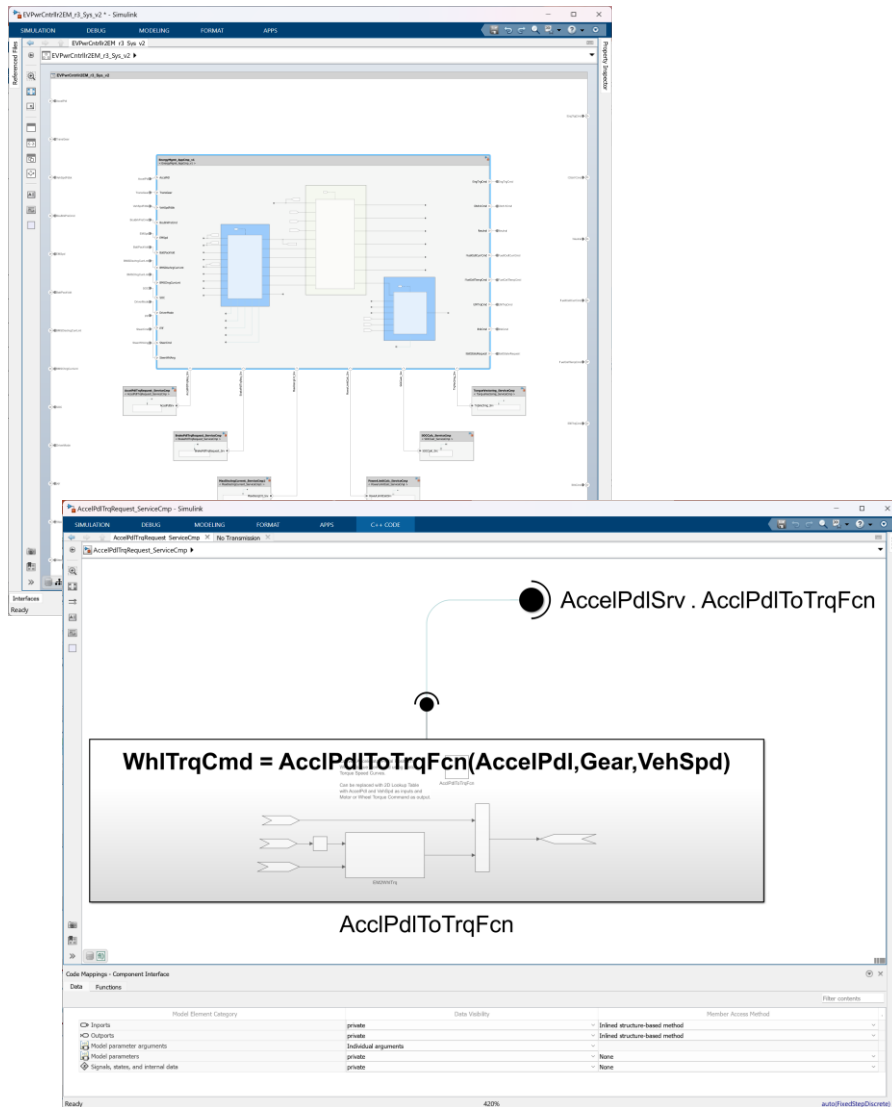
PreAlgoFcn Body

Call out to AccelPdlTrqReq_Srv Service Function

...\\MAC_SOA_Work\work\03_VCU_Models_ExpPart2a\CodeGen\EnergyMgmt_AppCmp_v1_ert_rtw\EnergyMgmt_AppCmp_v1.cpp Ln 1376 Col 37

OK Help

Generate C++ code that is agnostic to any platform



Service Component

Code Generation Report

Find: Match Case

Current model: **AccelPdTrqRequest_ServiceCmp** ▼

Content

- Summary
- Subsystem Report
- Code Interface Report
- Traceability Report
- Static Code Metrics Report
- Code Replacements Report
- Coder Assumptions

Code

- ▼ Main file
 - ert_main.cpp
- ▼ Model files
 - AccelPdTrqRequest_ServiceCmp.cpp
 - AccelPdTrqRequest_ServiceCmp.h
 - AccelPdTrqRequest_ServiceCmp_privat
 - AccelPdTrqRequest_ServiceCmp_types.
- ▼ Shared files
 - AccelPdTrqReq_SrvIf.t.h
 - const_params.cpp
 - look1_binlca.cpp
 - look1_binlca.h
 - rt_defines.h
 - rtwtypes.h

Actual Service Function

```

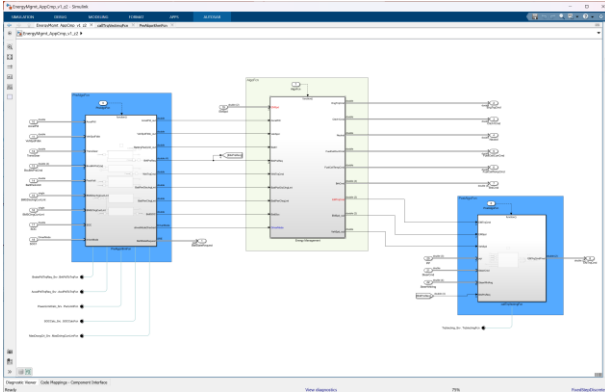
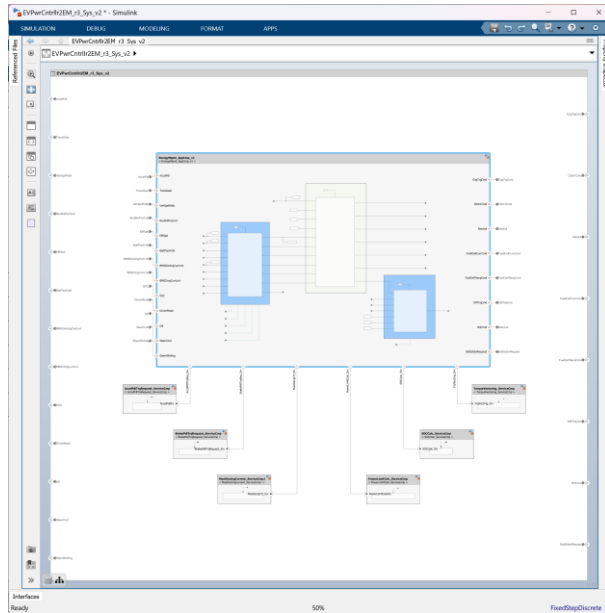
AccelPdTrqRequest_ServiceCmp.cpp
13 // 1. Execution efficiency
14 // 2. RAM efficiency
15 // 3. ROM efficiency
16 // Validation result: Not run
17 //
18 #include "AccelPdTrqRequest_ServiceCmp.h"
19 #include "rtwtypes.h"
20 #include <cmath>
21 #include "look1_binlca.h"
22 #include "AccelPdTrqRequest_ServiceCmp_private.h"
23
24 // Model step function
25 void AccelPdTrqRequest_ServiceCmp::AccelPdToTrqFcn(real_T rtu_AccelPd1, real_T
26 rtu_Gear, real_T rtu_VehSpd, real_T *rty_WhlTrqCmd)
27 {
28     real_T rtb_Gain3;
29     real_T rtb_MathFunction1_tmp;
30     real_T rtb_MaxEMTrqVsSpd;
31     int32_T rtb_MathFunction1;
32     int32_T tmp;
33     UNUSED_PARAMETER(rtu_Gear);
34
35     // Outputs for Function Call SubSystem: '<Root>/AccelPdToTrqFcn'
36     // Gain: '<S13>/Gain1' incorporates:
37     //   Gain: '<S13>/Gain2'
38     //   Gain: '<S13>/Gain3'
39     // SignalConversion generated from: '<S1>/VehSpd'
40
41     rtb_MathFunction1_tmp = 3.0581039755351682 * rtu_VehSpd * 3.32;
42
43     // Look1_Binlca_MaxEMTrqVsSpd incorporates:

```

...t2a\CodeGen\AccelPdTrqRequest_ServiceCmp_ert_rtw\AccelPdTrqRequest_ServiceCmp.cpp Ln 35 Col 38

AP: Generate C++ Code for Adaptive AUTOSAR Platform

Application Component



Code Generation Report

Find: Match Case

Current model: EnergyMgmt_AppCmp_v1_z2 ▼

EnergyMgmt_AppCmp_v1_z2.cpp 🔍 Search

1589 ProvidedPort->EMTrqCmd.Send(arrMsgData);
 1590 }
 1591
 1592 // Model step function for TID3
 1593 void EnergyMgmt_AppCmp_v1_z2::PreAlgoFcn() // Explicit Task: PreAlgoFcn
 1594 {
 1595 ara::core::Array<double, 4 > arrayArgBrkPd1ToTrqFcn;
 1596 ara::core::Future<proxy::methods::AcclPd1ToTrqFcn::Output>
 1597 AcclPd1ToTrqFcnFuture;
 1598 ara::core::Future<proxy::methods::BrkPd1ToTrqFcn::Output> BrkPd1ToTrqFcnFuture;
 1599 ara::core::Future<proxy::methods::MaxDchrgCurrLimFcn::Output>
 1600 MaxDchrgCurrLimFcnFuture;
 1601 ara::core::Future<proxy::methods::PwrLimitFcn::Output> PwrLimitFcnFuture;
 1602 ara::core::Future<proxy::methods::SOCCalcFcn::Output> SOCCalcFcnFuture;
 1603 proxy::methods::AcclPd1ToTrqFcn::Output callOutput;
 1604 proxy::methods::BrkPd1ToTrqFcn::Output callOutput_1;
 1605 proxy::methods::MaxDchrgCurrLimFcn::Output callOutput_2;
 1606 proxy::methods::PwrLimitFcn::Output callOutput_0;
 1607 proxy::methods::SOCCalcFcn::Output callOutput_3;
 1608 std::shared_ptr<ara::core::Result<proxy::methods::AcclPd1ToTrqFcn::Output>>
 1609 AcclPd1ToTrqFcnResultPtr;
 1610 std::shared_ptr<ara::core::Result<proxy::methods::BrkPd1ToTrqFcn::Output>>
 1611 BrkPd1ToTrqFcnResultPtr;
 1612 std::shared_ptr<ara::core::Result<proxy::methods::MaxDchrgCurrLimFcn::Output>>
 1613 MaxDchrgCurrLimFcnResultPtr;
 1614 std::shared_ptr<ara::core::Result<proxy::methods::PwrLimitFcn::Output>>
 1615 PwrLimitFcnResultPtr;
 1616 std::shared_ptr<ara::core::Result<proxy::methods::SOCCalcFcn::Output>>
 1617 SOCCalcFcnResultPtr;

Content

- Summary
- Subsystem Report
- Code Interface Report
- Traceability Report
- Static Code Metrics Report
- Eliminated Blocks
- Code Replacements Report
- Coder Assumptions

Code

- Model files
 - EnergyMgmt_AppCmp_v1_z2.cpp
 - EnergyMgmt_AppCmp_v1_z2.h
 - EnergyMgmt_AppCmp_v1_z2_private.h
 - EnergyMgmt_AppCmp_v1_z2_types.h
- Shared files
 - binsearch_u32d_prevIdx.cpp
 - binsearch_u32d_prevIdx.h
 - complex_types.h
 - const_params.cpp
 - intrap2d_la.cpp
 - intrap2d_la.h
 - look1_binlca.cpp
 - look1_binlca.h

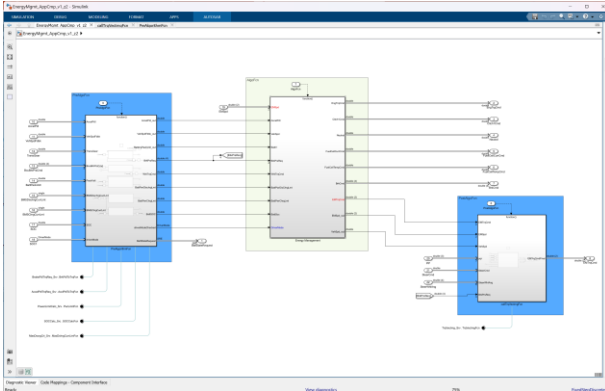
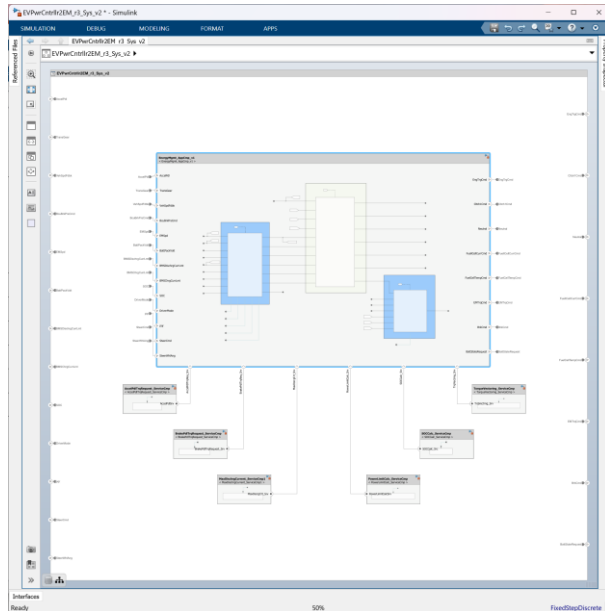
PreAlgoFcn Body

...MAC2024_DevShare\03_VCU_Models_ExpPart2b\CodeGen\EnergyMgmt_AppCmp_v1_z2_autosar_adaptive\EnergyMgmt_AppCmp_v1_z2.cpp Ln 1596 Col 3

OK Help

AP: Generate C++ Code for Adaptive AUTOSAR Platform

Application Component



Code Generation Report

Find: Match Case

Current model: EnergyMgmt_AppCmp_v1_z2 ▾

EnergyMgmt_AppCmp_v1_z2.cpp 🔍 Search

Content

- Summary
- Subsystem Report
- Code Interface Report
- Traceability Report
- Static Code Metrics Report
- Eliminated Blocks
- Code Replacements Report
- Coder Assumptions

Code

- Model files
 - EnergyMgmt_AppCmp_v1_z2.cpp
 - EnergyMgmt_AppCmp_v1_z2.h
 - EnergyMgmt_AppCmp_v1_z2_private.h
 - EnergyMgmt_AppCmp_v1_z2_types.h
- Shared files
 - binsearch_u32d_prevIdx.cpp
 - binsearch_u32d_prevIdx.h
 - complex_types.h
 - const_params.cpp
 - intrap2d_la.cpp
 - intrap2d_la.h
 - look1_binlca.cpp
 - look1_binlca.h

```

1650
1651 if (AccelPd1TrqReq_Srv) {
1652 // RootInportFunctionCallGenerator generated from: '<Root>/PreAlgoFcn' incorporates:
1653 // SubSystem: '<Root>/PreAlgorithmFcn'
1654
1655 // FunctionCaller: '<S2>/Acc1Pd1ToTrqFcn Caller'
1656 Acc1Pd1ToTrqFcnFuture = AccelPd1TrqReq_Srv->Acc1Pd1ToTrqFcn
1657 (EnergyMgmt_AppCmp_v1_z2_B.EventReceive,
1658 EnergyMgmt_AppCmp_v1_z2_B.EventReceive2,
1659 EnergyMgmt_AppCmp_v1_z2_B.EventReceive1);
1660
1661 // End of Outputs for RootInportFunctionCallGenerator generated from: '<Root>/PreAlgoFcn'
1662 // Retrieve result on method Acc1Pd1ToTrqFcn's completion
1663 Acc1Pd1ToTrqFcnResultPtr = std::make_shared< ara::core::Result<proxy::
1664 methods::Acc1Pd1ToTrqFcn::Output> >(Acc1Pd1ToTrqFcnFuture.GetResult());
1665
1666 // Check if method Acc1Pd1ToTrqFcn completed successfully and returned valid results
1667 if (Acc1Pd1ToTrqFcnResultPtr->HasValue()) {
1668 // Retrieve return arguments from method Acc1Pd1ToTrqFcn's Result container
1669 callOutput = Acc1Pd1ToTrqFcnResultPtr->Value();
1670
1671 // FunctionCaller: '<S2>/Acc1Pd1ToTrqFcn Caller'
1672 EnergyMgmt_AppCmp_v1_z2_B.Acc1Pd1ToTrqFcnCaller = callOutput.Wh1TrqCmd;
1673 }
1674 }
1675
1676 // RootInportFunctionCallGenerator generated from: '<Root>/PreAlgoFcn' incorporates:
1677 // SubSystem: '<Root>/PreAlgorithmFcn'

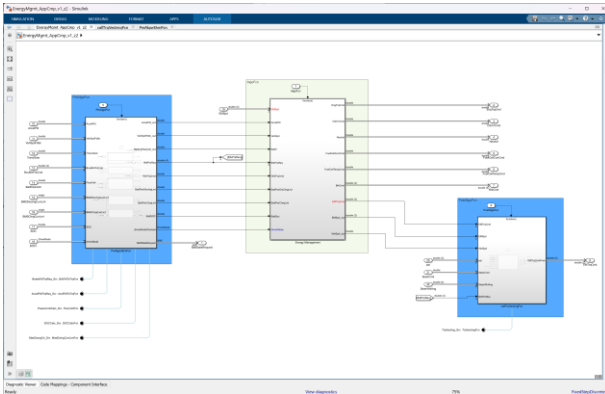
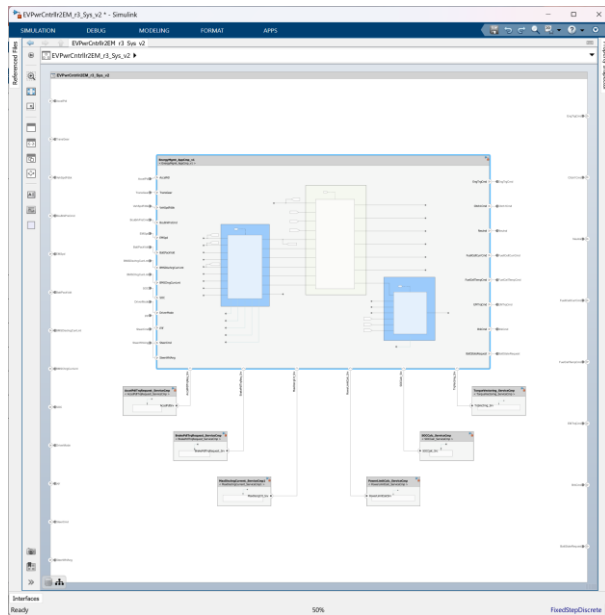
```

Call out to
AccelPd1TrqReq_Srv
Service Function

..MAC2024_DevShare\03_VCU Models_ExpPart2b\CodeGen\EnergyMgmt_AppCmp_v1_z2_autosar_adaptive\EnergyMgmt_AppCmp_v1_z2.cpp Ln 1669 Col 5

OK Help

AP: Generate C++ Code for Adaptive AUTOSAR Platform



Application Component

Code Generation Report

Find: Match Case

Current model: EnergyMgmt_AppCmp_v1_z2 ▾

Content

- Summary
- Subsystem Report
- Code Interface Report
- Traceability Report
- Static Code Metrics Report
- Eliminated Blocks
- Code Replacements Report
- Coder Assumptions

Code

- EnergyMgmt_AppCmp_v1_z2
- EnergyMgmt_AppCmp_v1_z2
- EnergyMgmt_AppCmp_v1_z2
- EnergyMgmt_AppCmp_v1_z2
- EnergyMgmt_AppCmp_v1_z2

- Other files
 - PosixExecutor.hpp
- Other files
 - main.cpp
- ARA files
 - MachineManifest.arxml
 - accelpdltrreq_srvif_com
 - accelpdltrreq_srvif_pro
 - brakepdltrreq_srvif_con
 - brakepdltrreq_srvif_pro
 - impl_type_drivermode.h
 - impl_type_rt_array_doub

main.cpp Search

```

9  #include <ara/core/initialization.h>
10 #include <ara/core/result.h>
11 #include <ara/core/string_view.h>
12 #include <ara/exec/execution_client.h>
13 #include <ara/log/common.h>
14 #include <ara/log/logger.h>
15 #include <ara/log/log_stream.h>
16 #include "EnergyMgmt_AppCmp_v1_z2.h"
17
18 /* main() handles the following: */
19 /* - Instantiates the model object and owns its memory. */
20 /* - Reports the Execution state to ARA */
21 /* - Calls the model's initialize and terminate functions. */
22 /* - Creates an executor instance to schedule the periodic step functions */
23 /* - A timer that is set to the base rate is created in the executor */
24 /* - The step functions are added to the executor and run */
25 /* based on their sample periods */
26 int32_t main() {
27     /* Used to control the flow in case of error in any api's used. */
28     bool bProceed{true};
29     /* Used to decide whether ara function clusters has been initialized. */
30     bool bAraInitialized{true};
31     /* ara function cluster init. */
32     const ara::core::Result<void> initStatus{ara::core::Initialize()};
33
34     if (!initStatus.HasValue()) {
35         bProceed = false;
36         bAraInitialized = false;
37     } /* if */
38
39     if (bAraInitialized) {
40         ara::log::Logger &araLog{ara::log::CreateLogger(

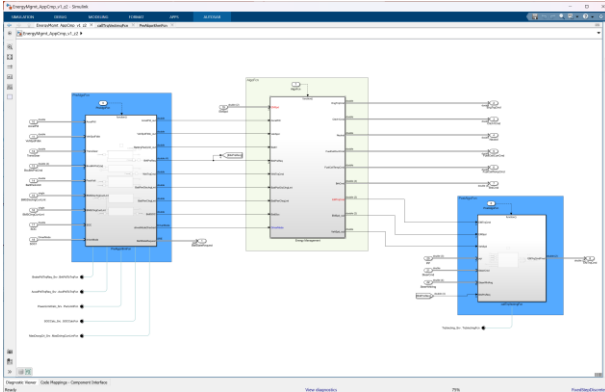
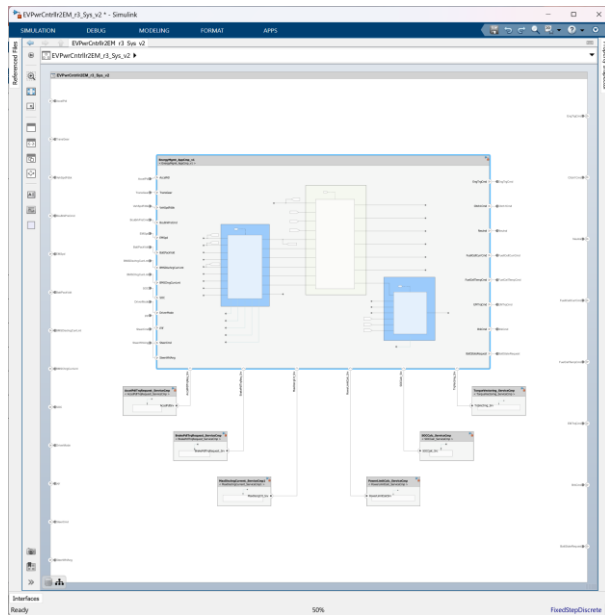
```

...orks\Documents\00_AE_Info_Content\AE_Share\MAC2024_DevShare\03_VCU_Models_ExpPart2b\CodeGen\EnergyMgmt_AppCmp_v1_z2_autosar_adaptive\main.cpp Ln 27 Col 6

Main.cpp / Application

OK Help

AP: Generate C++ Code for Adaptive AUTOSAR Platform



Application Component

Main.cpp / Application
 Adding tasks for Main Application executor

Code Generation Report

Find: Match Case

Current model: **EnergyMgmt_AppCmp_v1_z2**

Content

- Summary
- Subsystem Report
- Code Interface Report
- Traceability Report
- Static Code Metrics Report
- Eliminated Blocks
- Code Replacements Report
- Coder Assumptions

Code

- EnergyMgmt_AppCmp_v1_z2
- Other files
 - PosixExecutor.hpp
- Other files
 - main.cpp
- ARA files
 - MachineManifest.xml
 - accelpdltrreq_srvif_com
 - accelpdltrreq_srvif_pro
 - brakepdltrreq_srvif_com
 - brakepdltrreq_srvif_pro
 - impl_type_drivermode.h
 - impl_type_rt_array_doub
 - impl_type_rt_array_doub
 - impl type rt array doub

main.cpp

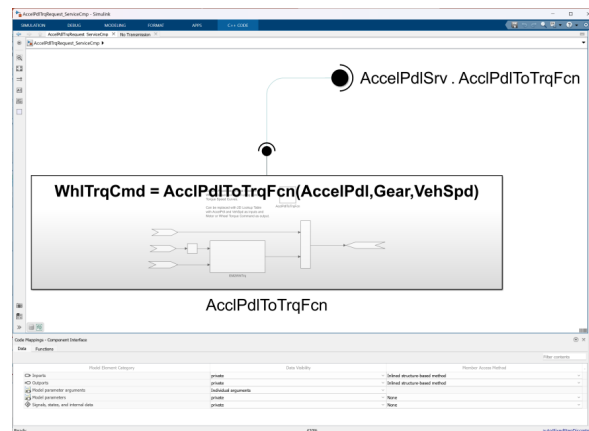
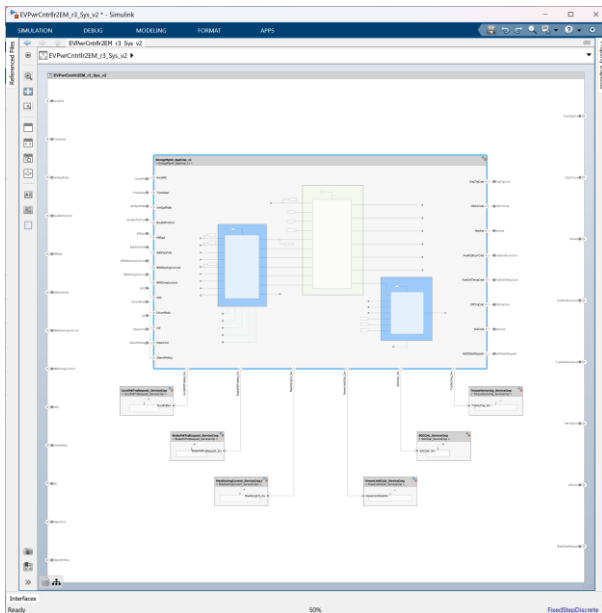
```

82     fcnExecutor.setBaseRateInSeconds(std::chrono::duration<double>(baseRate));
83
84     /* Register periodic step functions in the executor. */
85     fcnExecutor.addPeriodicEvent(
86         [&EnergyMgmt_AppCmp_v1_z2_Obj, &araLog]() {
87             try {
88                 EnergyMgmt_AppCmp_v1_z2_Obj.PreAlgoFcn();
89             } catch (std::exception const &e) {
90                 araLog.LogError() << "Error executing step: " << e.what();
91             }
92         },
93         1);
94     fcnExecutor.addPeriodicEvent(
95         [&EnergyMgmt_AppCmp_v1_z2_Obj, &araLog]() {
96             try {
97                 EnergyMgmt_AppCmp_v1_z2_Obj.AlgoFcn();
98             } catch (std::exception const &e) {
99                 araLog.LogError() << "Error executing step: " << e.what();
100            }
101        },
102        1);
103     fcnExecutor.addPeriodicEvent(
104         [&EnergyMgmt_AppCmp_v1_z2_Obj, &araLog]() {
105             try {
106                 EnergyMgmt_AppCmp_v1_z2_Obj.PostAlgoFcn();
107             } catch (std::exception const &e) {
108                 araLog.LogError() << "Error executing step: " << e.what();
109             }
110        },
111        1);
            
```

...Share\MAC2024_DevShare\03_VCU Models_ExpPart2b\CodeGen\EnergyMgmt_AppCmp_v1_z2_autosar_adaptive\main.cpp Ln 107 Col 35

OK Help

AP: Generate C++ Code for Adaptive AUTOSAR Platform



Service Component

Code Generation Report

Find: Match Case

Current model: **AccelPdITrqRequest_ServiceCmp**

Content

- Summary
- Subsystem Report
- Code Interface Report
- Traceability Report
- Static Code Metrics Report
- Code Replacements Report
- Coder Assumptions

Code

- ▼ Model files
 - AccelPdITrqRequest_ServiceCmp.cpp
 - AccelPdITrqRequest_ServiceCmp.h
 - AccelPdITrqRequest_ServiceCmp_priv
 - AccelPdITrqRequest_ServiceCmp_typ
- ▼ Shared files
 - complex_types.h
 - const_params.cpp
 - look1_binlcapw.cpp
 - look1_binlcapw.h
 - rt_defines.h
- ▼ Interface files
 - AccelPdITrqRequest_ServiceCmp_Exec
 - AccelPdITrqRequest_ServiceCmp_Ser
 - AccelPdITrqRequest_ServiceCmp_cor

AccelPdITrqRequest_ServiceCmp.cpp Search

```

19 #include <stdint.h>
20 #include "look1_binlcapw.h"
21 #include "AccelPdITrqRequest_ServiceCmp_private.h"
22
23 // Model step function
24 void AccelPdITrqRequest_ServiceCmp::AccIPdToTrqFcn(double AccelPdI, double Gear,
25 double VehSpd, double *WhlTrqCmd)
26 {
27     double rtb_Gain3;
28     double rtb_MathFunction1_tmp;
29     double rtb_MaxEMTrqVsSpd;
30     int32_t rtb_MathFunction1;
31     int32_t tmp;
32     UNUSED_PARAMETER(Gear);
33
34     // Outputs for Function Call SubSystem: '<Root>/AccIPdToTrqFcn'
35     // Gain: '<S13>/Gain1' incorporates:
36     //   Gain: '<S13>/Gain2'
37     //   Gain: '<S13>/Gain3'
38     // SignalConversion generated from: '<S1>/VehSpd'
39
40     rtb_MathFunction1_tmp = 3.0581039755351682 * VehSpd * 3.32;
41
42     // Lookup_n-D: '<S6>/MaxEMTrqVsSpd' incorporates:
43     //   Gain: '<S13>/Gain1'
44
45     rtb_MaxEMTrqVsSpd = look1_binlcapw(rtb_MathFunction1_tmp,
46 &rtCP_MaxEMTrqVsSpd_bp01Data[0], &rtCP_MaxEMTrqVsSpd_tableData[0], 13U);
47
          
```

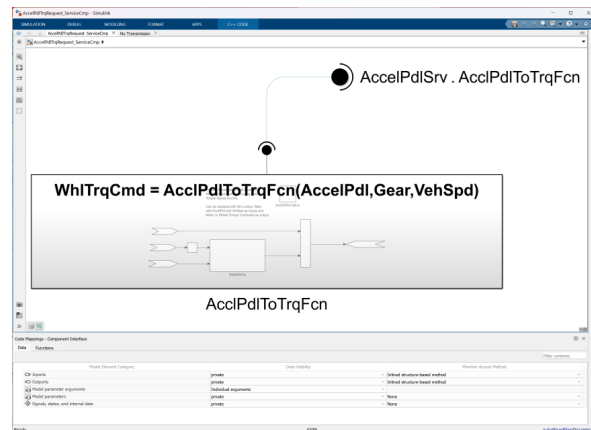
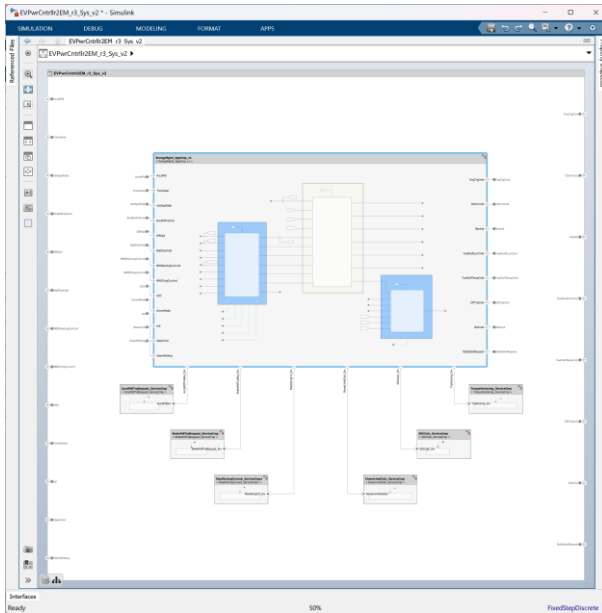
Actual Service Function

...t2b\CodeGen\AccelPdITrqRequest_ServiceCmp_autosar_adaptive\AccelPdITrqRequest_ServiceCmp.cpp Ln 19 Col 3

OK Help

40

AP: Generate C++ Code for Adaptive AUTOSAR Platform



Service Component

Current model: AccelPdTrqRequest_ServiceCmp ▾

Code Generation Report

Find: Match Case

Current model: **AccelPdTrqRequest_ServiceCmp ▾**

- AccelPdTrqRequest_ServiceCmp.h
- AccelPdTrqRequest_ServiceCmp_priv
- AccelPdTrqRequest_ServiceCmp_typ
- ▼ Shared files
 - complex_types.h
 - const_params.cpp
 - look1_binlcapw.cpp
 - look1_binlcapw.h
 - rt_defines.h
- ▼ Interface files
 - AccelPdTrqRequest_ServiceCmp_Exec
 - AccelPdTrqRequest_ServiceCmp_Ser
 - AccelPdTrqRequest_ServiceCmp_cor
 - AccelPdTrqRequest_ServiceCmp_dat
 - AccelPdTrqRequest_ServiceCmp_inte
- ▼ Other files
 - PosixExecutor.hpp
 - main.cpp
- ▼ Other files
 - MachineManifest.xml
 - accelpdtrreq_srvif_common.h
 - accelpdtrreq_srvif_skeleton.h
 - accelpdtrreq_srvif_skeleton_impl.h

main.cpp Search

```

77  /* AccelPdTrqRequest_ServiceCmp::AccelPdToTrqFcn(); */
78  /* These asynchronous tasks are registered */
79  /* as callbacks in the model initialize function: */
80  /* AccelPdTrqRequest_ServiceCmp::initialize() */
81
82  /* Create an executor instance to schedule the periodic step functions. */
83  /* Whenever the period of a step function passes, the executor */
84  /* schedules that function to be executed on a thread. */
85  platform::runtime::Executor fcnExecutor;
86
87  /* Base rate is the time unit of a tick. */
88  constexpr double baseRate{0.200000};
89  fcnExecutor.setBaseRateInSeconds(std::chrono::duration<double>(baseRate));
90
91  araLog.LogVerbose() << "Starting periodic execution of step functions.";
92  #if defined(rtmSetStopRequested) && defined(rtmGetStopRequested)
93      fcnExecutor.run(
94          [&AccelPdTrqRequest_ServiceCmp_Obj]() {
95              return rtmGetStopRequested(
96                  AccelPdTrqRequest_ServiceCmp_Obj.getRTM());
97          },
98          araLog);
99  #else
100     fcnExecutor.run(araLog);
101  #endif
102  } /* if */
103
104  if (bProceed) {
105      try {

```

Ln 30 Col 1

Ln 85 Col 39

Main.cpp / Service

Adding Service to executor

OK Help

Key Take Aways

- We have the tools to help move to Service Oriented Architectures (SOA)
- We can generate C++ code for SOA based components
- We are going to show you a lot of information in this presentation
- We are here to help so please engage with us in your SOA type projects

MathWorks Resources

[Solutions Page: Software-Defined Vehicle](#)



[Solutions Page: Software Architecture](#)



Where can I find more information ?

[What Is Service-Oriented Architecture \(SOA\)?](#)

Service-Oriented Architecture (SOA) Search MathWorks.com

What Is Service-Oriented Architecture?

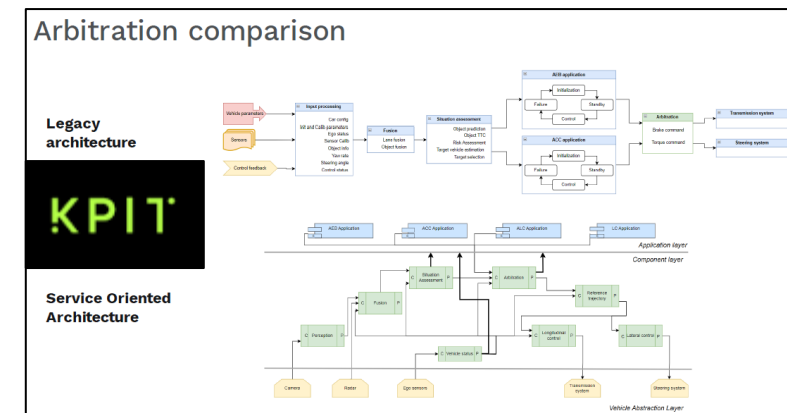
Service-oriented architecture (SOA) is a modern software architecture paradigm for building applications as a collection of modular units of software called services. In SOA, services are self-contained, modular, and loosely coupled. This approach enables you to build complex and distributed applications in which you can update individual components, in contrast to entire monolithic applications. A typical SOA software stack includes application software comprising services, platform services, and middleware. These services run on high-performance hardware or virtual machines.

Generalized SOA software stack.

[Using Model-Based Design to Develop SOA Applications for In-Vehicle OS](#)



[Service-Oriented Arbitration of ADAS Features with Model-Based Design](#)



Training and Consulting Services

Where can I get more help ?

The screenshot shows the MathWorks Self-Paced Online Courses interface. The course title is "System Composer Onramp". A green "Start course" button is visible, along with a progress bar at 0%. The page includes a "Share Course" link, "Share Certificate & Progress", and "Settings". A description states: "Become familiar with model-based systems engineering constructs in System Composer. You will construct a system architecture and trace to system requirements, create specialized views of the architecture, and link to Simulink to verify the behavior of your system." The "About this course" section lists: Format: Self-paced, Length: About 1 hour, Language: English. Prerequisites include "Simulink Onramp". The course is authored by Alisha Schor from MathWorks. The "Course modules" section includes "Course Overview" (Length: About 5 min, Preview the course), "Constructs", and "Requirements".

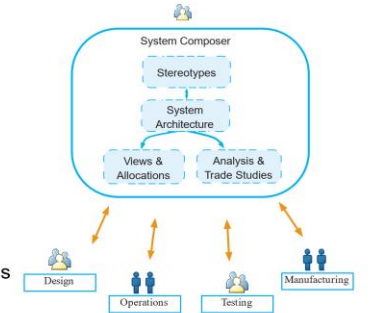
System Composer Onramp

System Composer for Architecture Modeling

After this 1-day training you will be able to:

- Identify the role of architectural modeling in Model-Based Design
- Create functional, logical, and physical architectures
- Customize existing components, ports and connections
- Analyze architecture models for early requirement validation
- Generate customized architecture visualizations

[See detailed course outline](#)



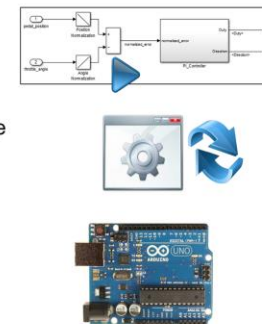
System Composer Course Details

Embedded Coder for Production Code Generation

Topics included in this 3-day course:

- Generated code structure and execution
- Code generation options and optimizations
- Integrating generated code with external code
- Generating code for multirate systems
- Customizing generated code
- Customizing data
- Deploying code

[See detailed course outline](#)



Embedded Coder Course Details

Training and Consulting Services



Code Generation for Classic AUTOSAR Software Components

After this two-day training you will be able to:

- Generate Simulink models from existing ARXML system descriptions
 - Configure Simulink models for AUTOSAR compliant code generation
 - Configure AUTOSAR communication elements in a Simulink model
 - Model AUTOSAR events in Simulink
 - Create calibration parameters
 - Model AUTOSAR variation points within software components
 - Import and export AUTOSAR compositions and software architectures
 - Model calls to basic software services
- [See detailed course outline](#)

1

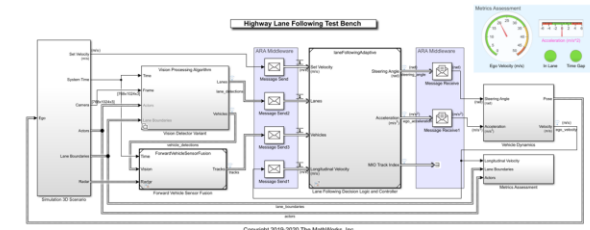
[Classic AUTOSAR Course Details](#)

Where can I get more help ?



AUTOSAR Adaptive Jumpstart Consulting Service

- Just enough Adaptive AUTOSAR
 - Learn Adaptive AUTOSAR elements relevant for MBD of application software
- Relevant exercises and workflows
 - Converting existing models to the Adaptive Platform
 - Choosing appropriate modeling constructs for Adaptive services
 - Developing, simulating, and testing Adaptive Applications
 - Generating and reviewing C++ AUTOSAR code
- Choose from 8-16hrs of coaching to fit your needs
 - AUTOSAR Adaptive Overview
 - Classic vs Adaptive
 - Adaptive platform architecture
 - Component modeling
 - Component code generation
 - Simulation, verification, and validation
 - Configuring for deployment
 - Measurement and calibration



AUTOSAR
Adaptive Platform


1-2 Day(s) / Onsite Instructor Led
Adaptive AUTOSAR Training

3







[AUTOSAR Adaptive Platform Jumpstart Course Details](#)

Training and Consulting Services

Where can I get more help ?



Seamlessly transition to Service-Oriented Architecture

					
Jumpstart	Migration Plan Development	Design SOA	Integrate Design Into SOA	Update Workflow for SOA MBD	Deploy Models to SOA Platforms
Jumpstart with hands-on workshop including available solutions and industry trends	Evaluate existing software development workflow and develop a migration plan	Architect and analyze a SOA using System Composer	Integrate detailed design of service applications into SOA model	Migration of existing development workflow to support SOA MBD workflow	Deployment of the models onto a chosen middleware platform such as Adaptive AUTOSAR and custom middleware

<https://www.mathworks.com/services/consulting/contact.html>

Technical Workshop on this topic – June 20th 1:30 to 3:30 PM EDT



If you have questions, you can reach out to me

Mark Danielsen
mdaniels@mathworks.com

Technical Workshop: Migration of a Monolithic Algorithm to Service-Oriented Architecture

Date & Time: June 20, 1:30PM – 3:30PM EDT

Overview: 2-hour interactive workshop at our Novi office to take a deep dive into the process of breaking apart a monolithic algorithm into services that can be reused.

MathWorks
**AUTOMOTIVE
CONFERENCE 2024**
North America

Thank you



© 2024 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [mathworks.com/trademarks](https://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.