

Evolution of Simulink Semantics for AUTOSAR Adaptive Applications

Sep 21, 2023 | Troy, Michigan

Jeff Harper



Are we in the era of Software-Defined Vehicle?

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



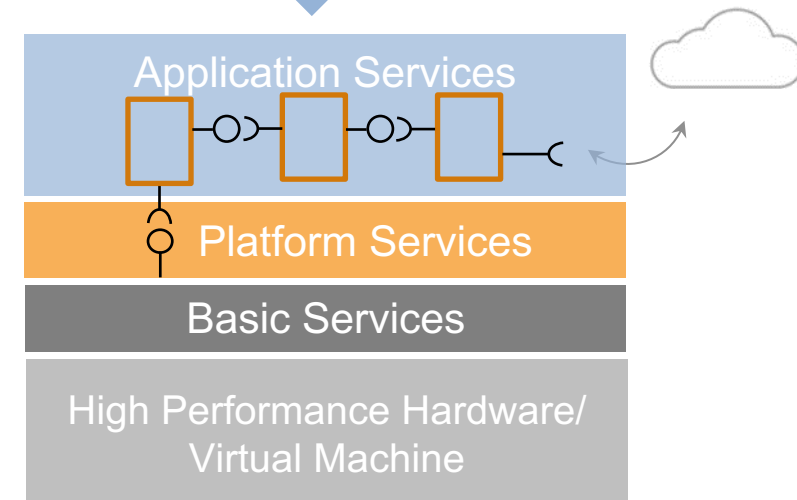
Brand-distinctive features and main value for the customer will come from Software

100110
001010
1001100010
001010
100110
001010
010010



SW updates

- Frequent
- Selective
- Over-the-air



Higher HW abstraction:
Service-oriented architectures

Agenda

- Software-defined vehicles and new architectures
- New Simulink Semantics
 - Generic Service Oriented Architecture (SOA)
 - SOA based AUTOSAR Adaptive applications
- Conclusions and key takeaways

Software-defined vehicles

Customer expectations

- Clean and Safe mobility
- Digital Life continuity

Technology & Innovation

- Electrification
- Autonomy
- Connectivity



monetize

Business opportunity

- App stores, SW features on demand
- SW services subscription plans

invest

demand

Centralization of computing and SOA



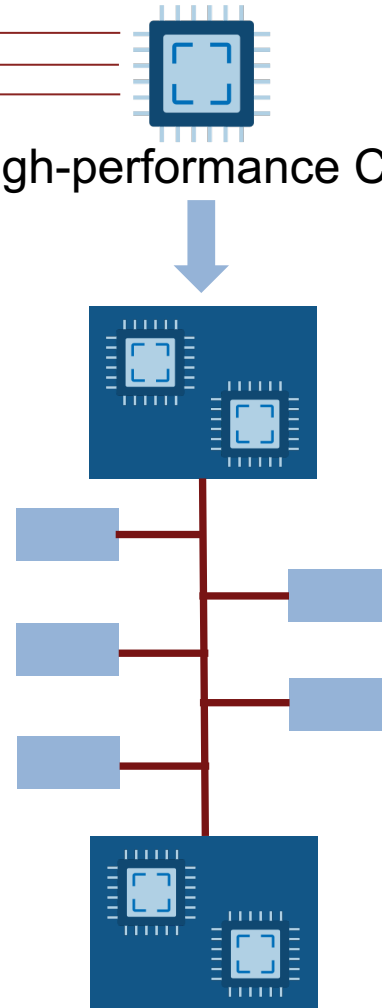
Exponential growth of SW features

100110
001010
010010

100110
001010
010010

100110
001010
010010

High-performance CPU/GPU

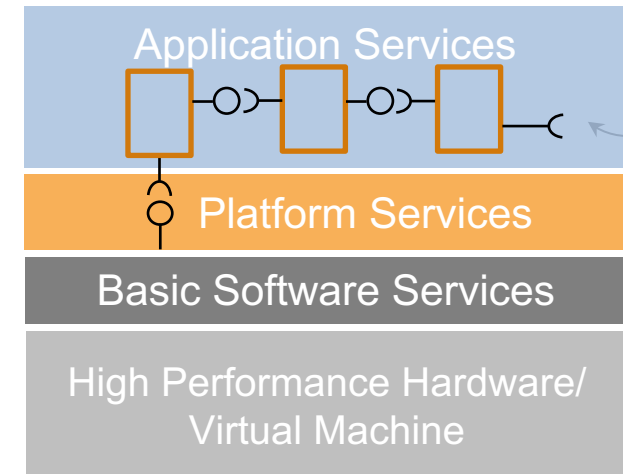


New E/E zonal architectures



SW updates

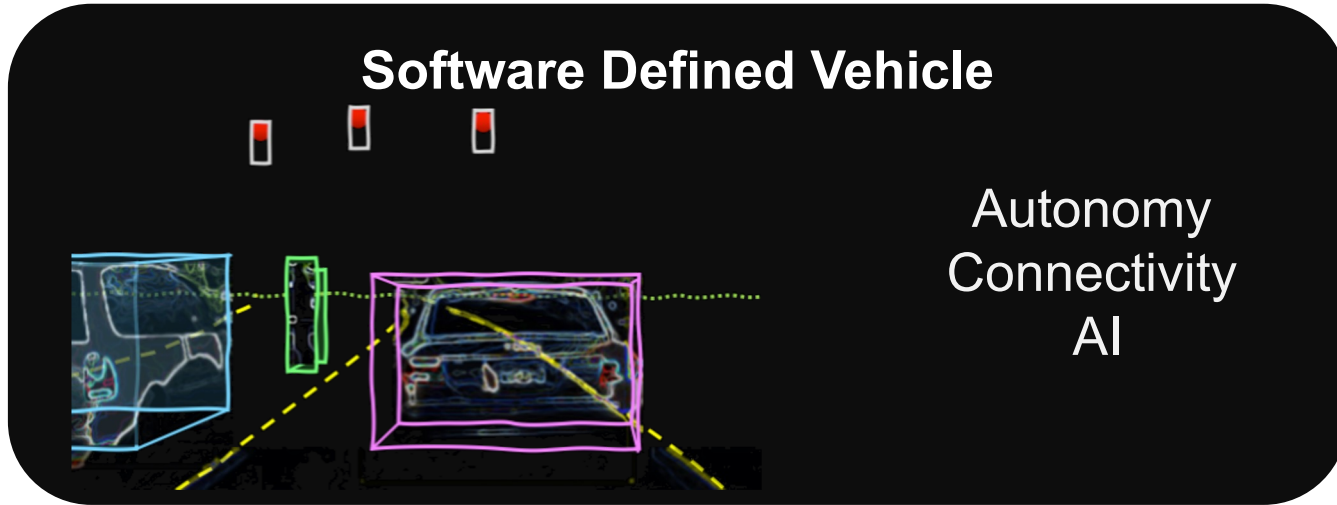
- Frequent
- Selective
- Over-the-air



Higher HW abstraction:
Service-oriented architectures

100110
001010
1001100010
001010
100110
001010
010010

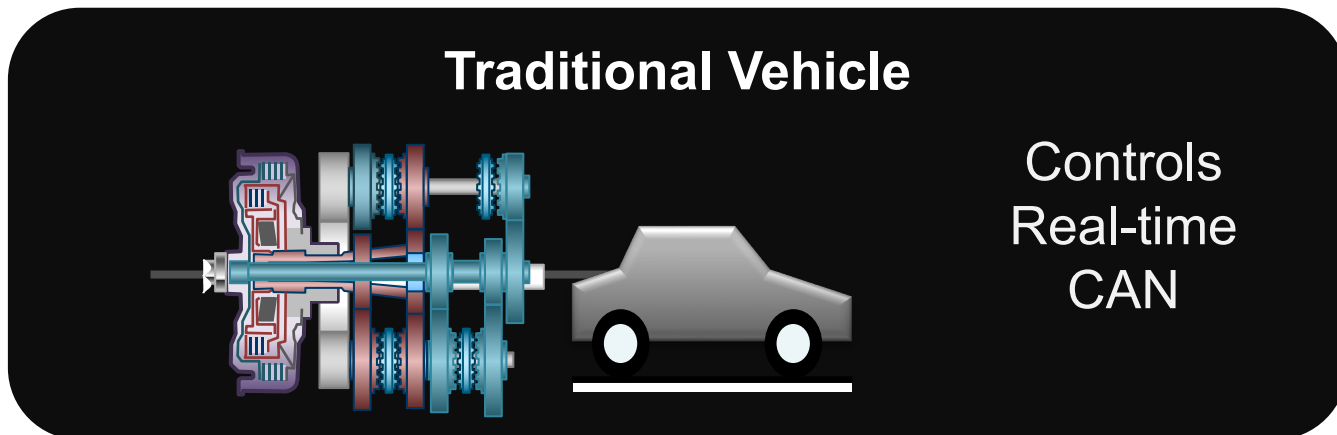
Automakers are increasingly building software in-house with SOA based design



Steering,
Braking



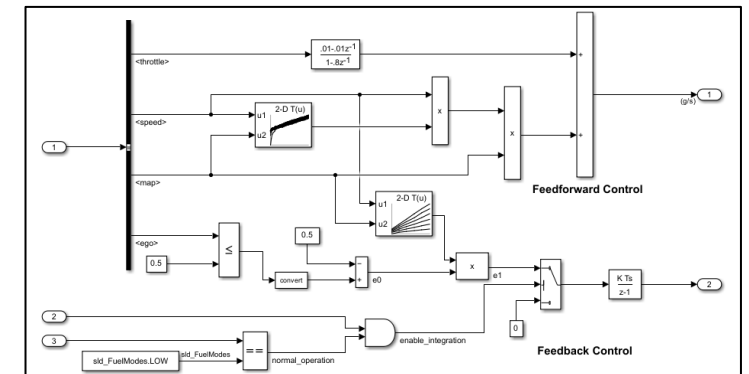
Speed,
Velocity



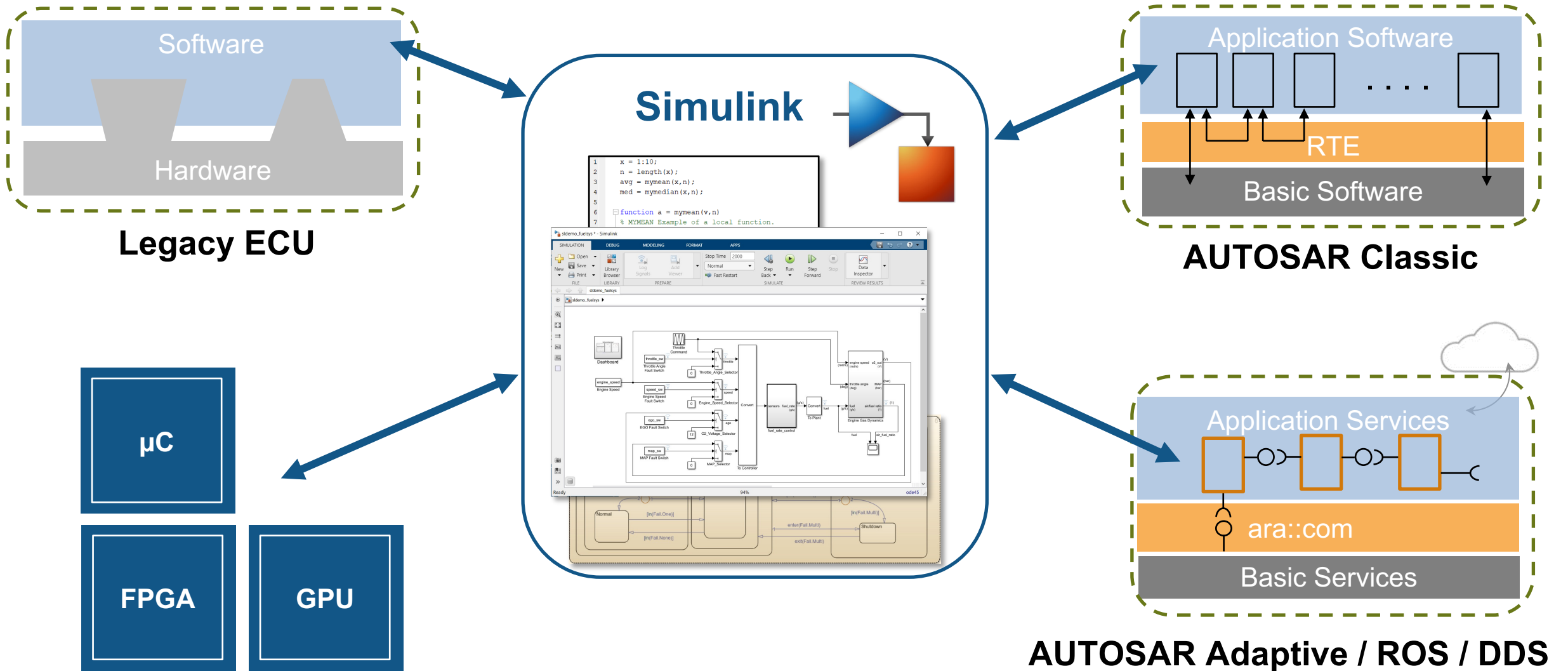
SOA based standards



Model-Based Design



Simulink: deploy software to different targets and standards

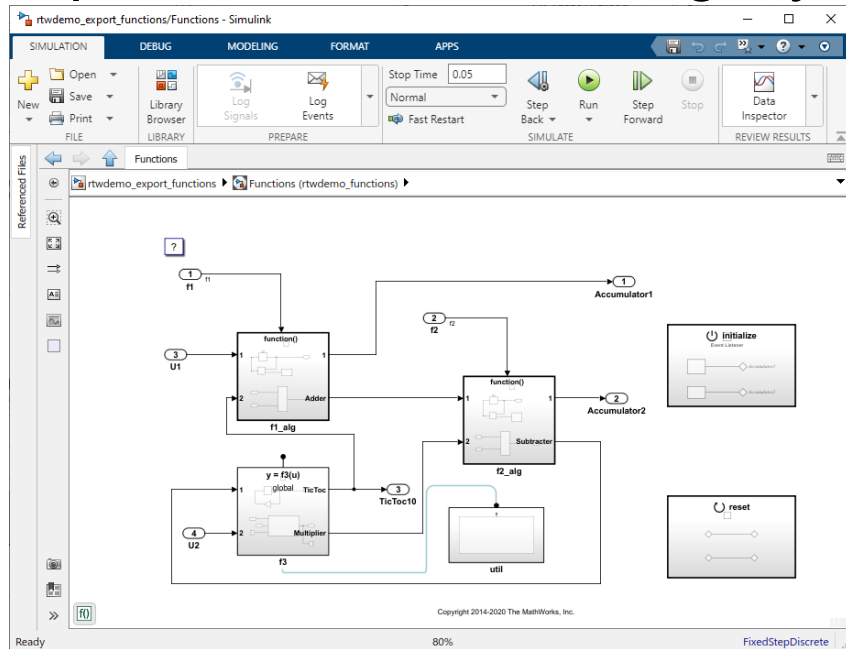


Agenda

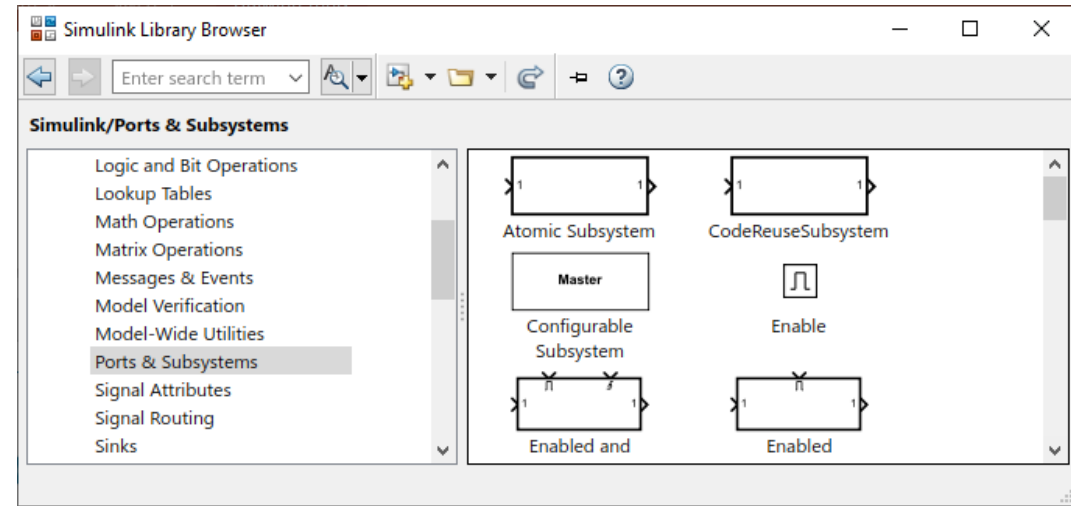
- Software-defined vehicles and new architectures (SOA)
- **New Simulink Semantics**
 - Generic SOA
 - SOA based AUTOSAR Adaptive applications
- Conclusions and key takeaways

Simulink Supports Exporting Callable Functions Well

- Export Function Modeling style



rtwdemo_export_functions

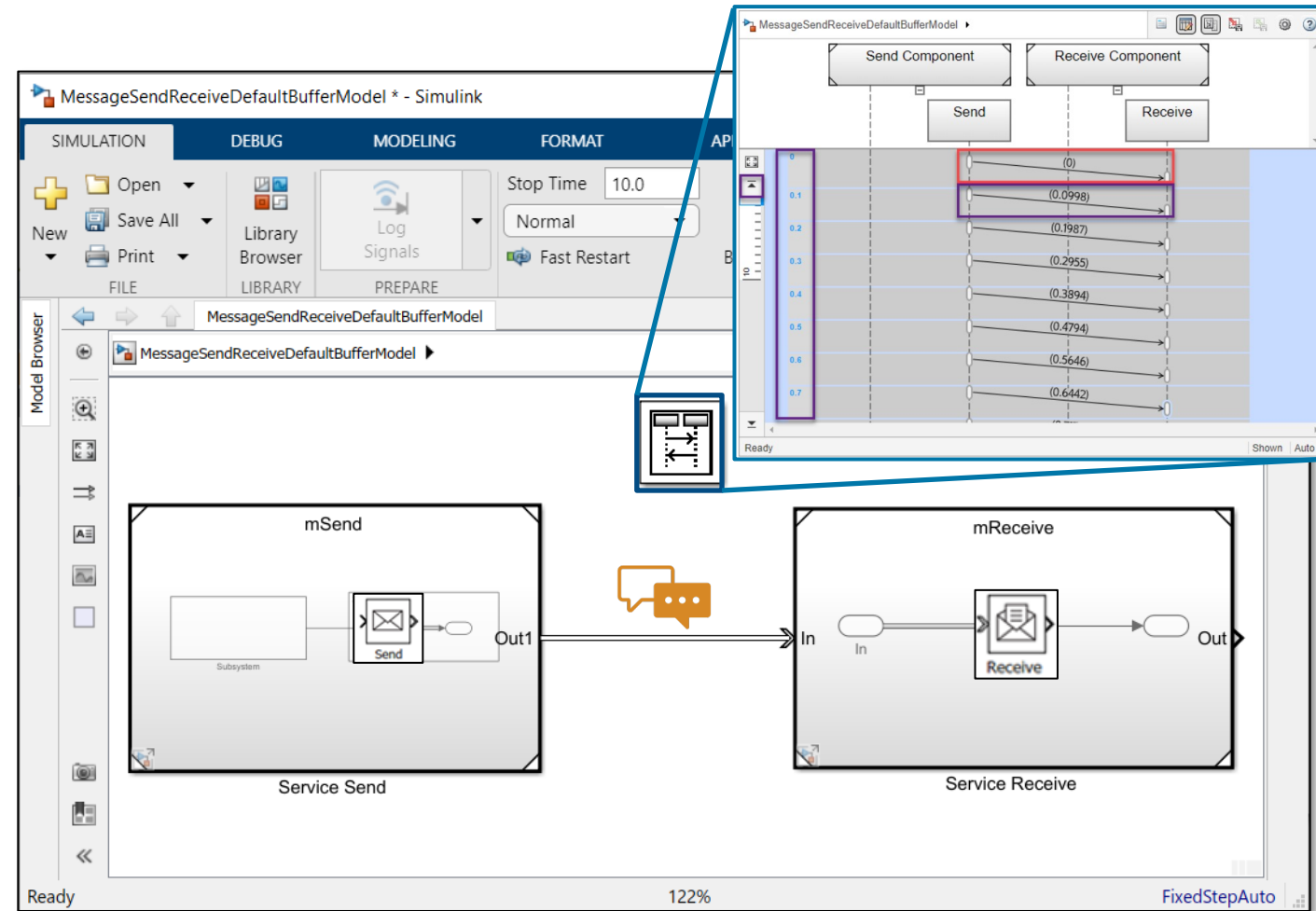
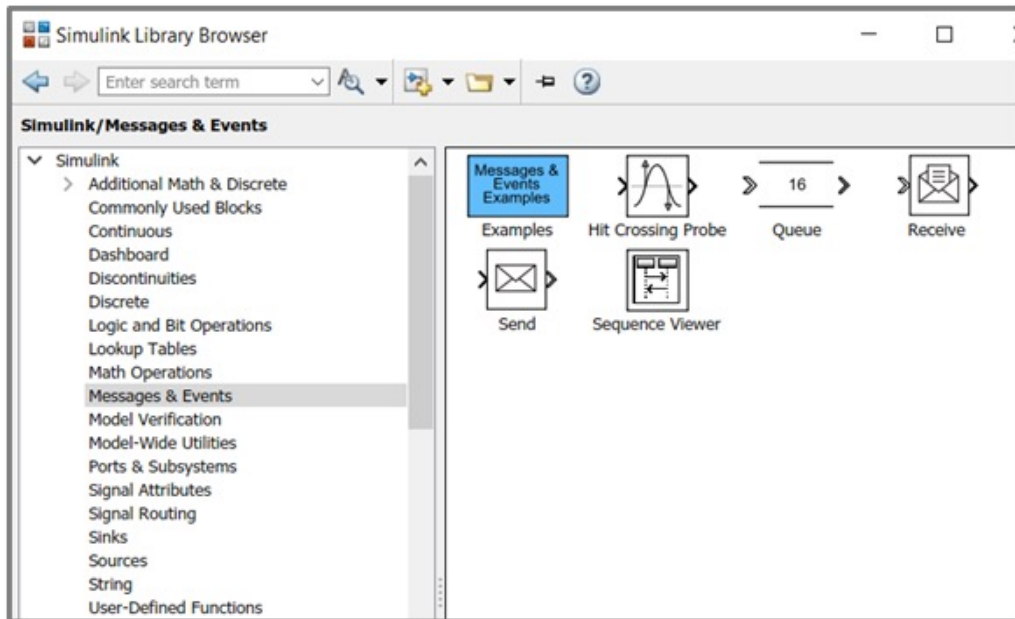
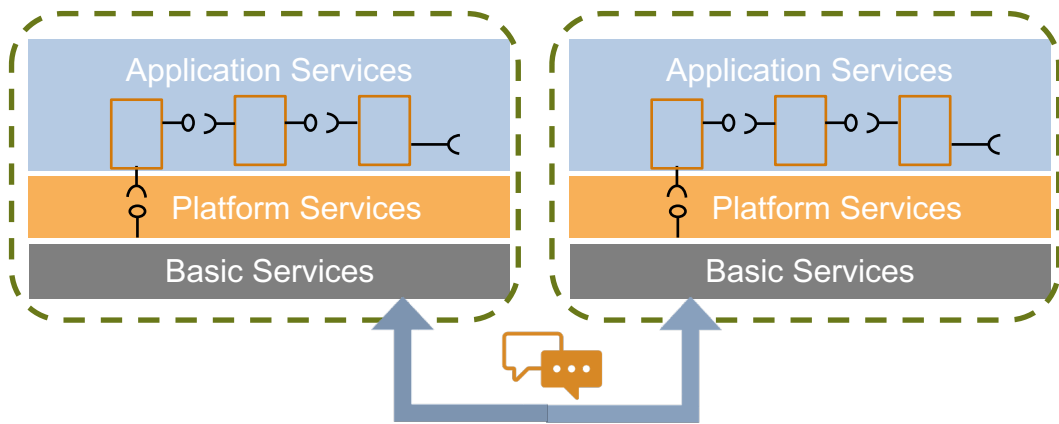


Ports and Subsystems

Agenda

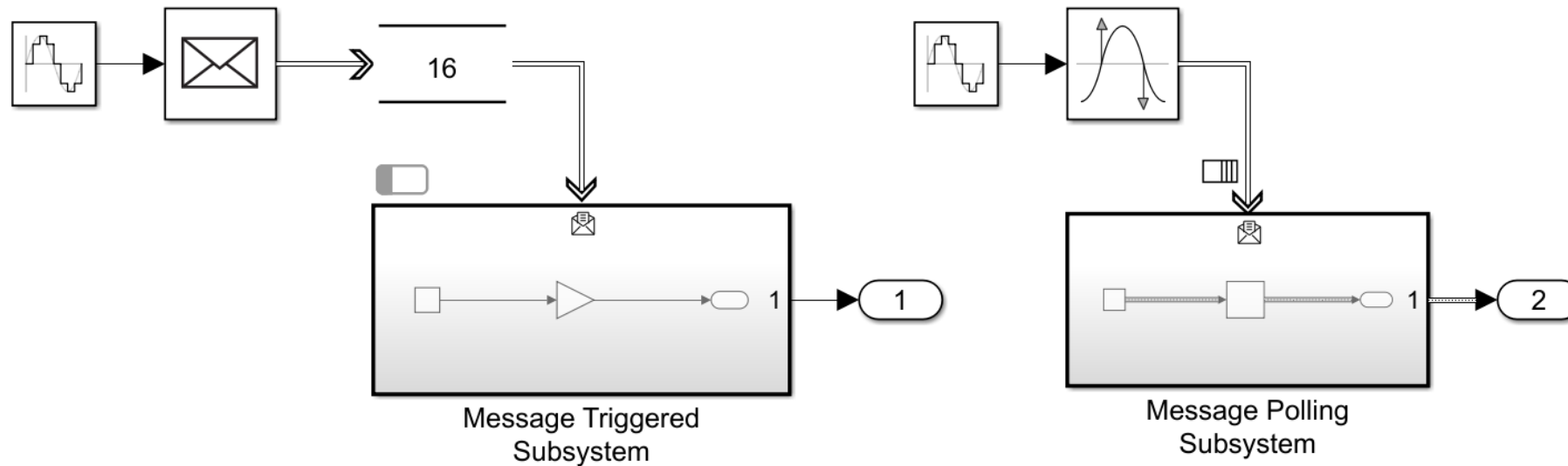
- Software-defined vehicles and new architectures (SOA)
- **New Simulink Semantics**
 - Generic SOA
 - SOA based AUTOSAR Adaptive applications
- Conclusions and key takeaways

Service-Oriented Behavior Modeling with Simulink Messages



You can model service-oriented communication using messages (Send/Receive)

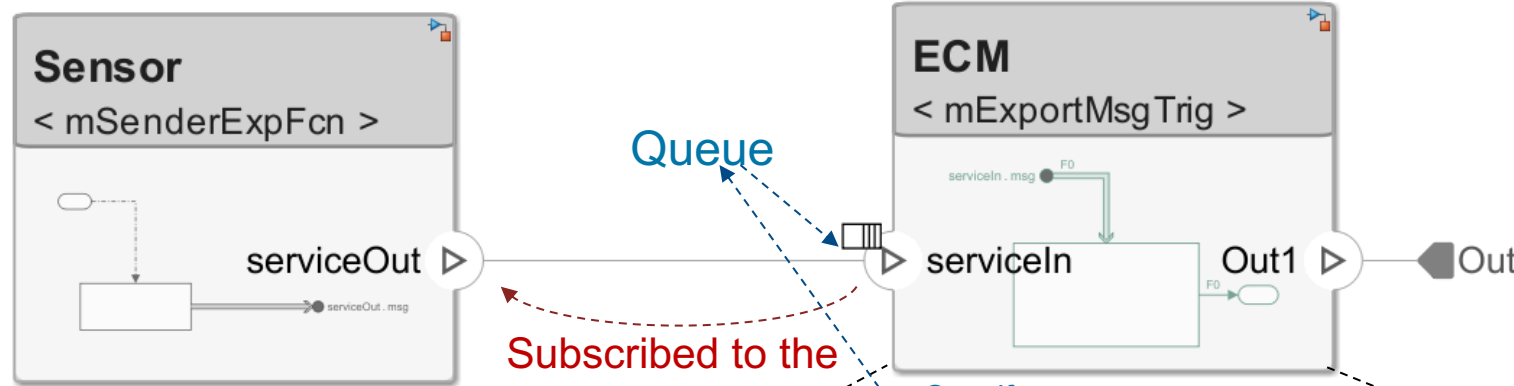
Service-Oriented Behavior Modeling with Message Triggered/Polling Subsystem



- New blocks to process messages by executing subsystem when message is available
- Model and generate code for components that are executed on message arrival

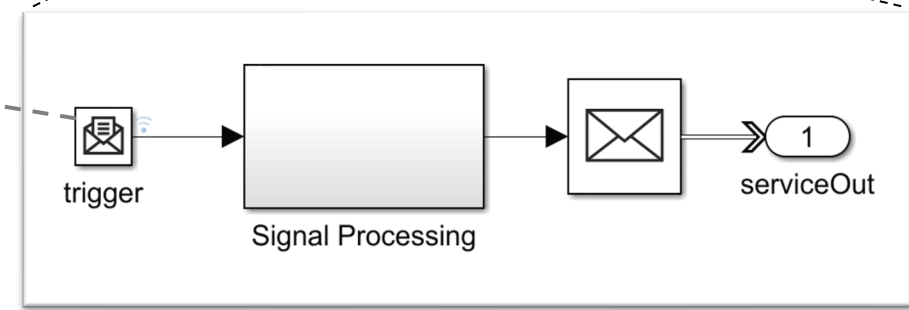
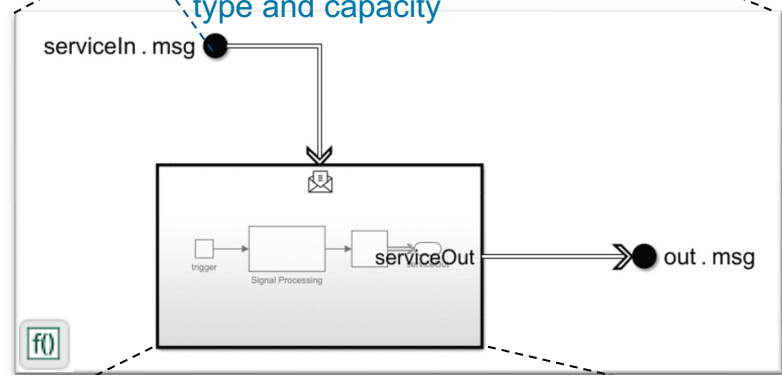
Service-Oriented Behavior Modeling with Message Triggered/Polling Subsystem

Event-Based Message Trigger



Subscribed to the message source

Specify queue type and capacity



Block Parameters: trigger

Trigger Port

Place this block in a subsystem or at the root level of a model to create a triggered or function-call system.

If the trigger type is "rising," "falling," or "either," placing this block at the root level of a model enables a Signal Attributes tab.

Main

Trigger type: message

Trigger time: on message available

Schedule as aperiodic partition

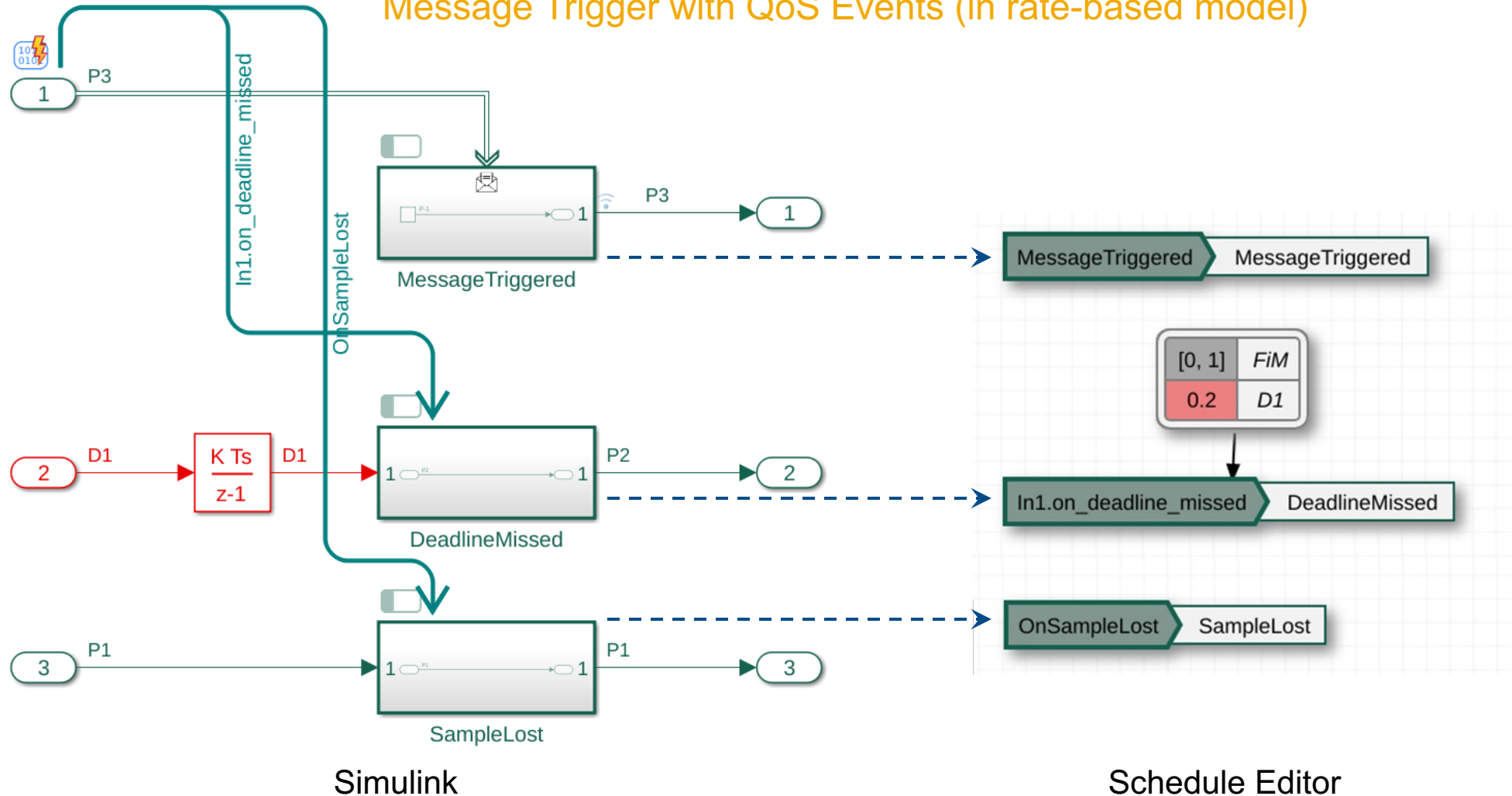
OK Cancel Help Apply

Use **Schedule Editor** or **Function Editor** to vary the execution order

- Code gen target supported:
- AUTOSAR Adaptive
 - ERT C++

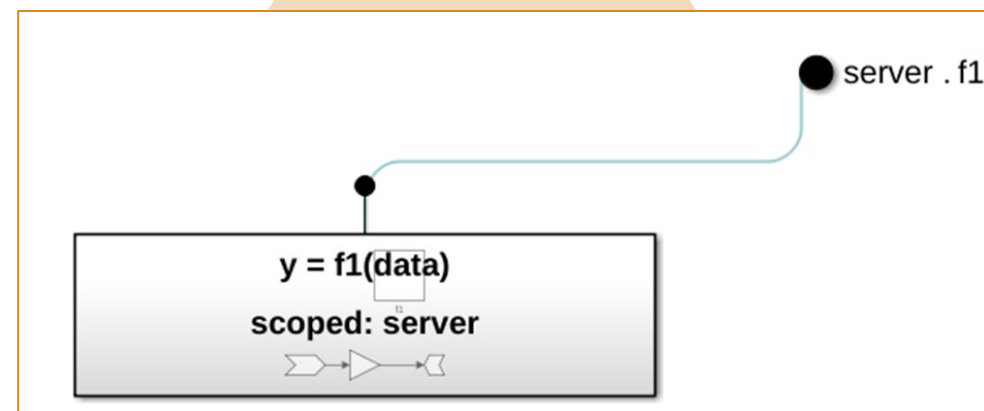
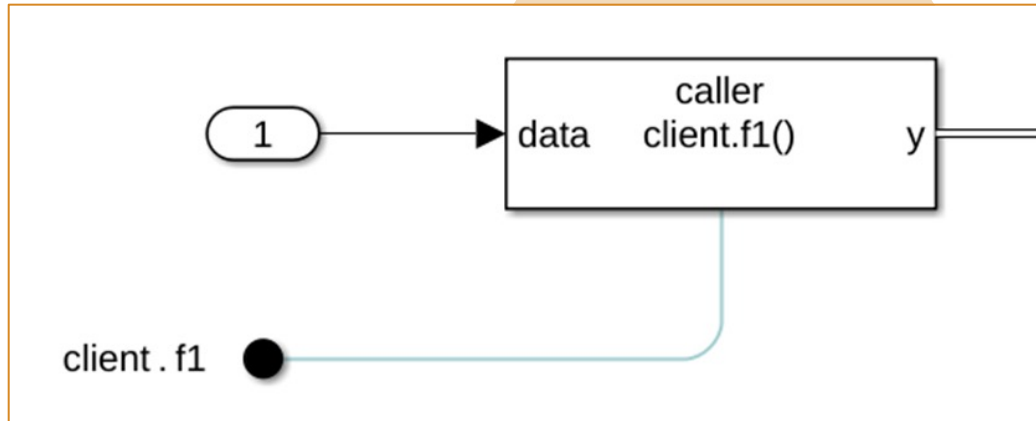
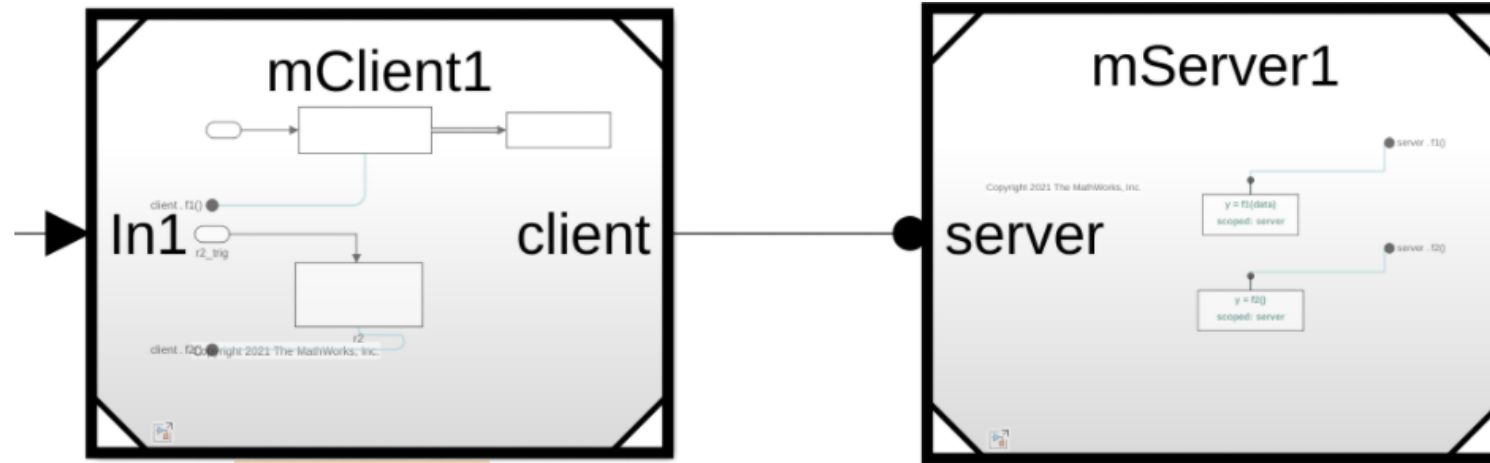
Service-Oriented Behavior Modeling with Message Triggered/Polling Subsystem

Message Trigger with QoS Events (in rate-based model)



Service-Oriented Behavior Modeling with Service-Based Functions

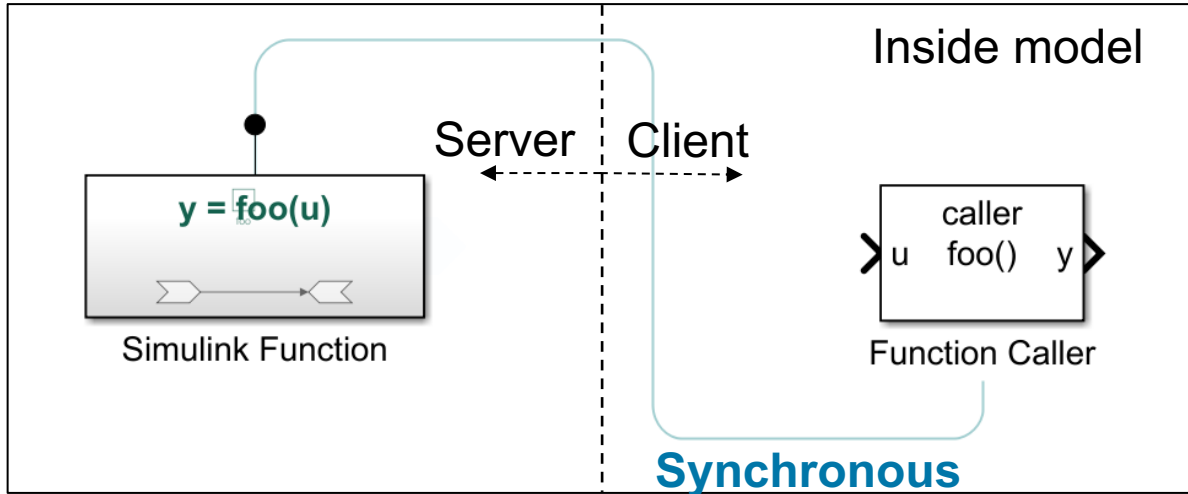
Function Ports for SOA



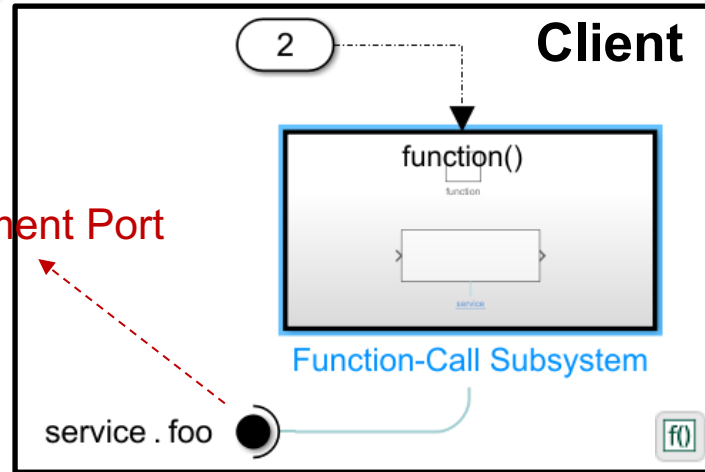
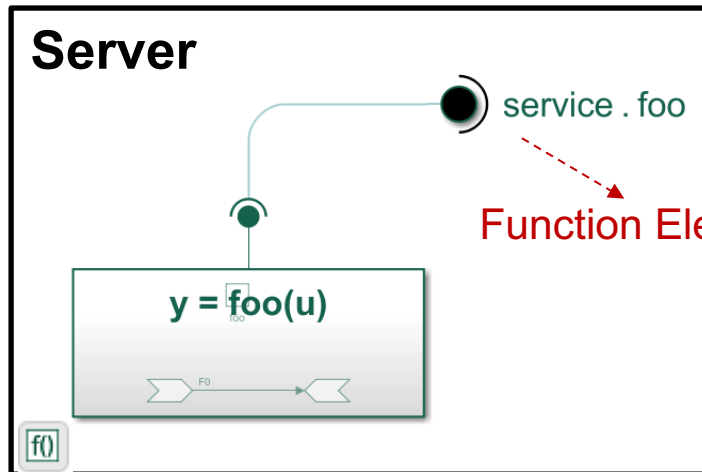
Model client and server components to facilitate data sharing using a functional interface between component models

Service-Oriented Behavior Modeling with Service-Based Functions

Behavior Model for Services Based on Simulink Function



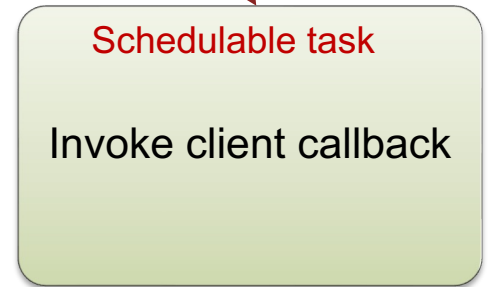
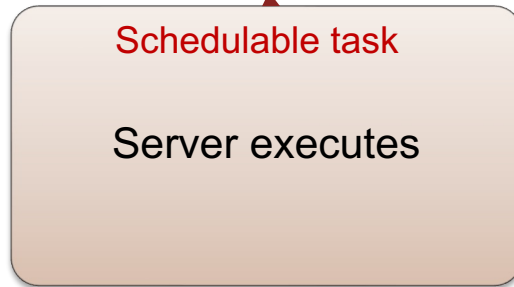
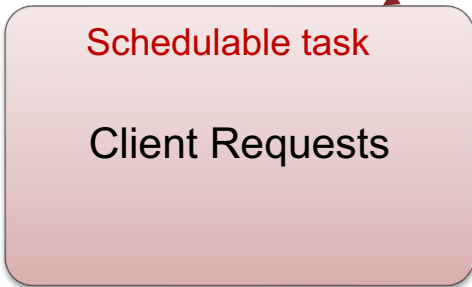
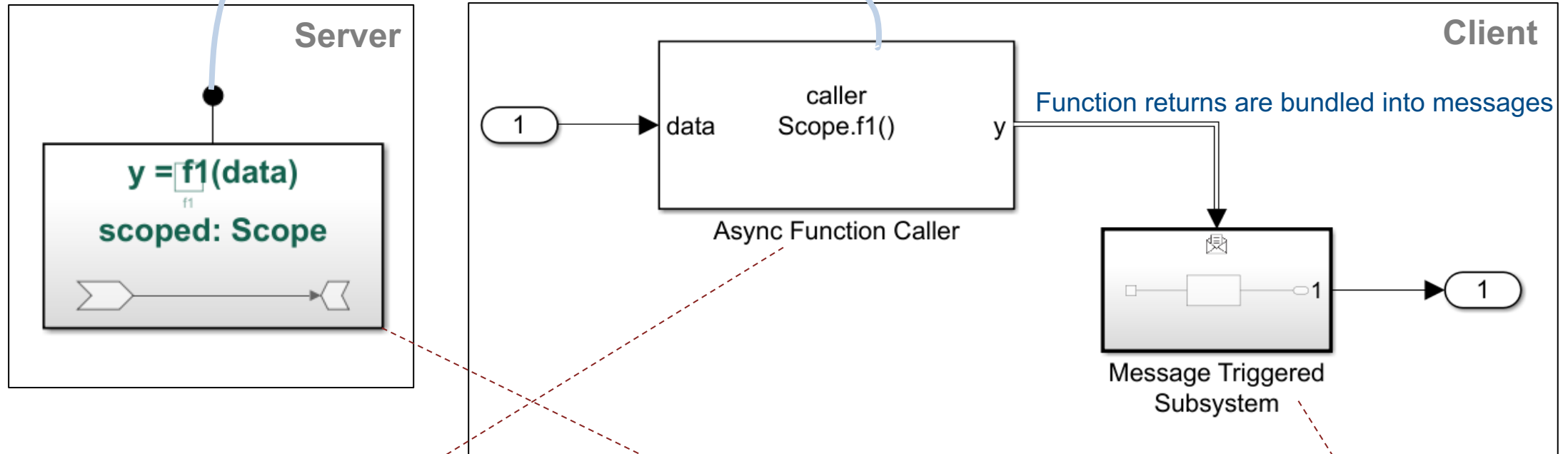
modularization



Across model

Service-Oriented Behavior Modeling with Service-Based Functions

Asynchronous Client-Server Function with Callback



Service-Oriented Behavior

SCHEDULE EDITOR

Manage Partitions | Order | Update Diagram | Save Model | Highlight | Arrange | Timing Legend | Layout

PARTITIONS | EXECUTION | MODEL | DISPLAY | VIEW

Events

Name
Client.ResponseCallback
Client.ResponseCallback1
Server.service.bar
Server.service.foo

```

    graph TD
      subgraph TopFlowchart
        S1[Server.service.foo] --> C1[Client_FunctionCall]
        C1 --> R1[Client.ResponseCallback]
      end
      subgraph BottomFlowchart
        S2[Server.service.bar] --> C2[Client_FunctionCall1]
        C2 --> R2[Client.ResponseCallback1]
      end
  
```

Order

Order	Name	Trigger
1	Server.service.foo	Server.s
2	Server.service.bar	Server.s
3	Client_FunctionCall	
4	Client_FunctionCall1	
5	Client.ResponseCallback	Client.R
6	Client.ResponseCallback1	Client.R

Drag to change the priority of a function

Legend

DATA CONNECTIONS

- Dependency
Source runs before destination
- > Delay
Destination runs before source
- 🔒 Prevent Delay
These connections are always dependencies
- 🔓 Allow Delay
These connections become delays if necessary

PARTITIONS

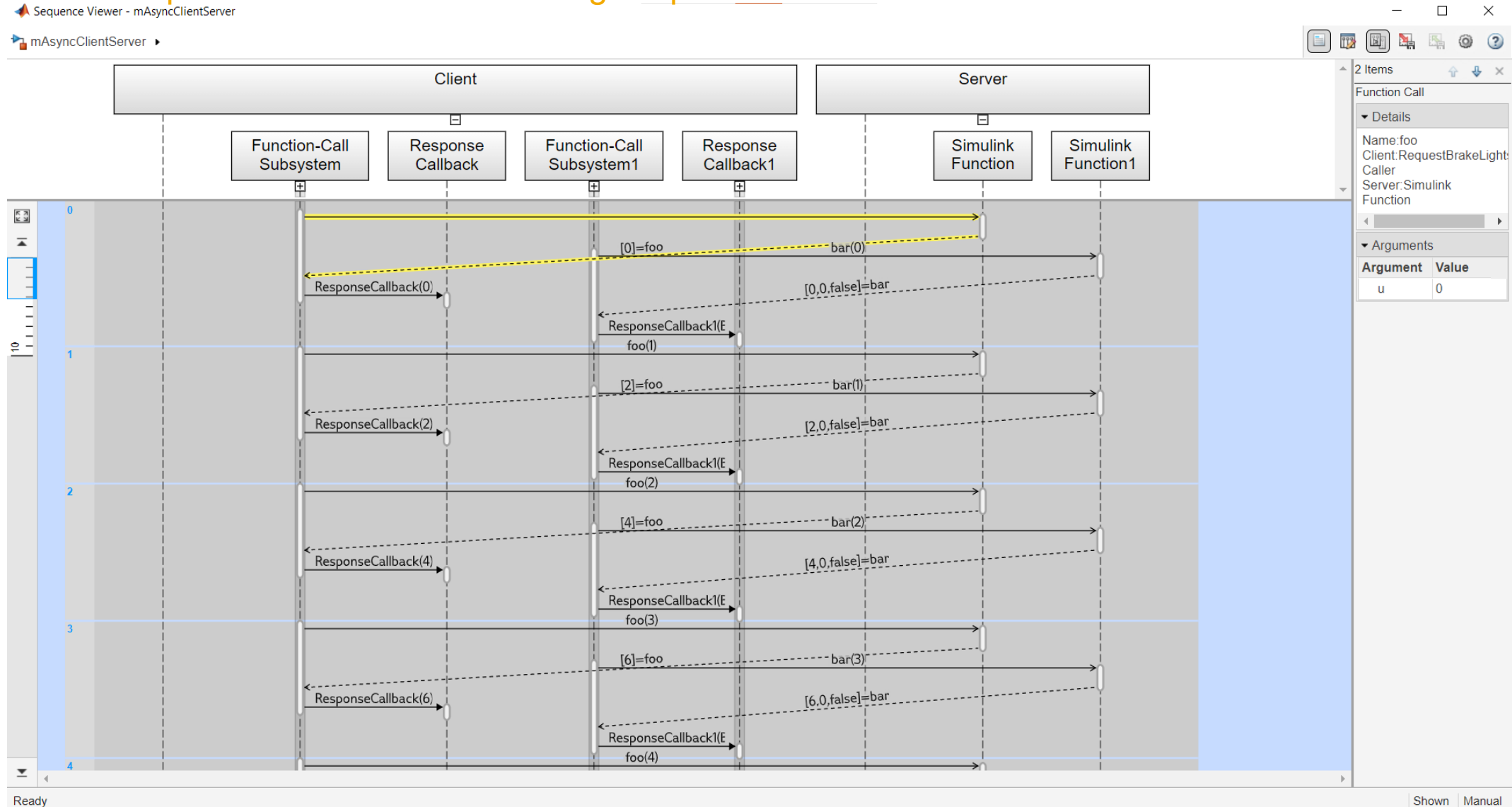
- 📄 Implicit
A partition automatically created from blocks that are not partitioned

Property Inspector

Partition	
Name	Server.service.foo
Rate	A
Type	Aperiodic exported function

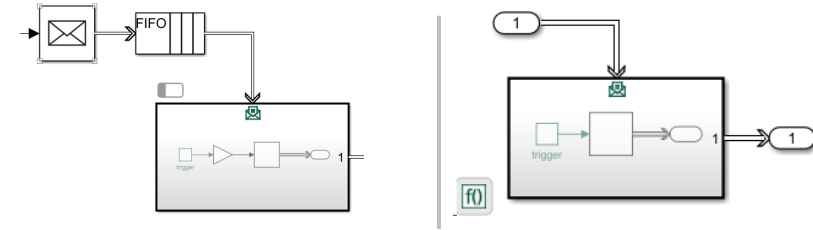
Service-Oriented Behavior Modeling

Visualize Sequence of Service Calls Using Sequence Viewer

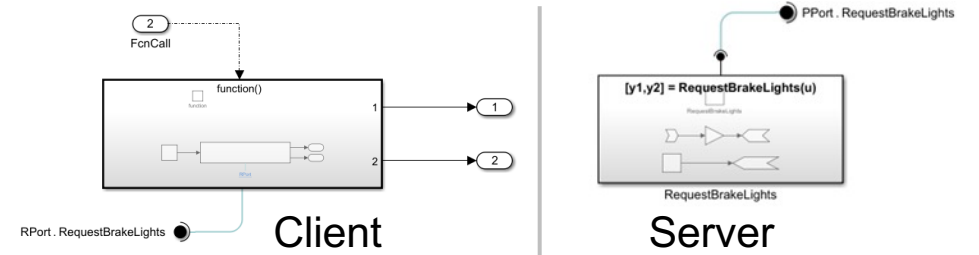


Behavior Modeling

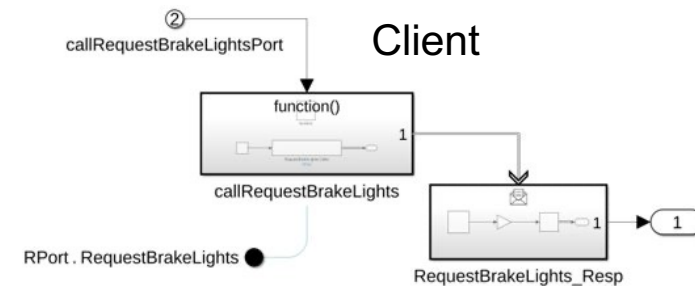
Message triggered subsystem



Synchronous client-server function port



Asynchronous client-server function port

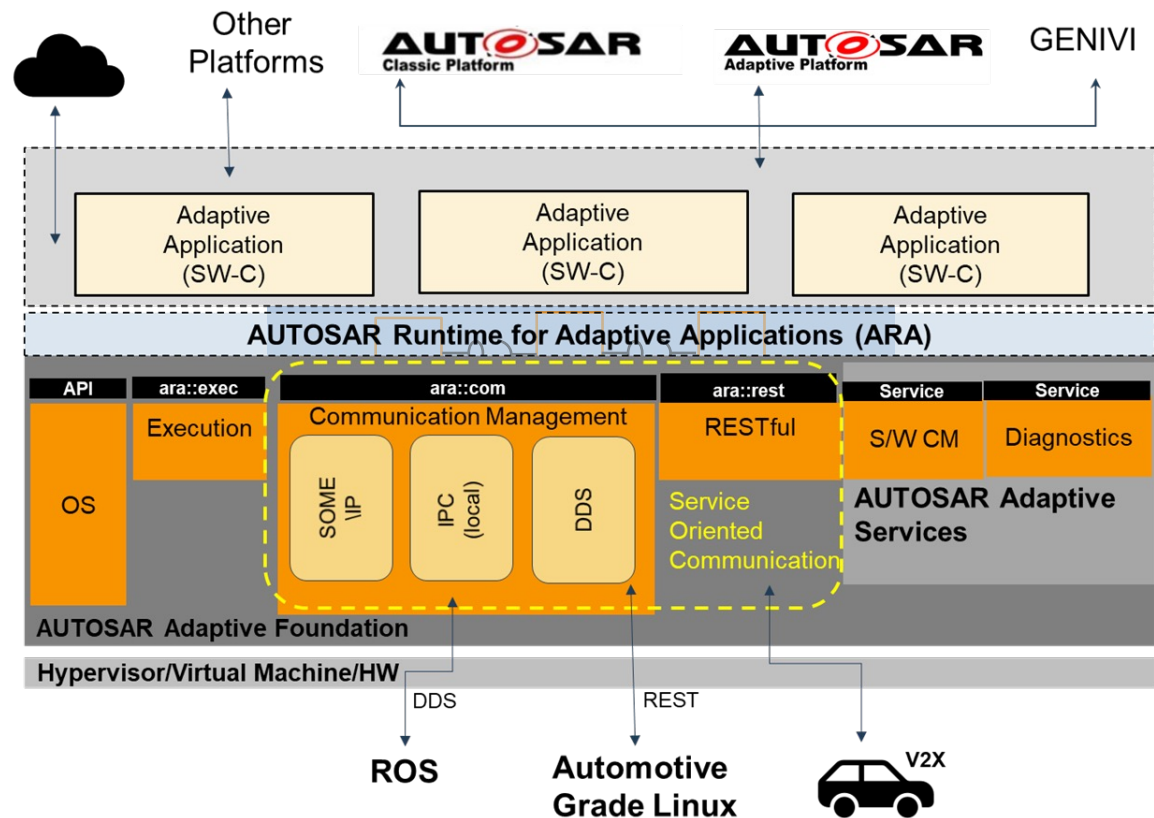


Agenda

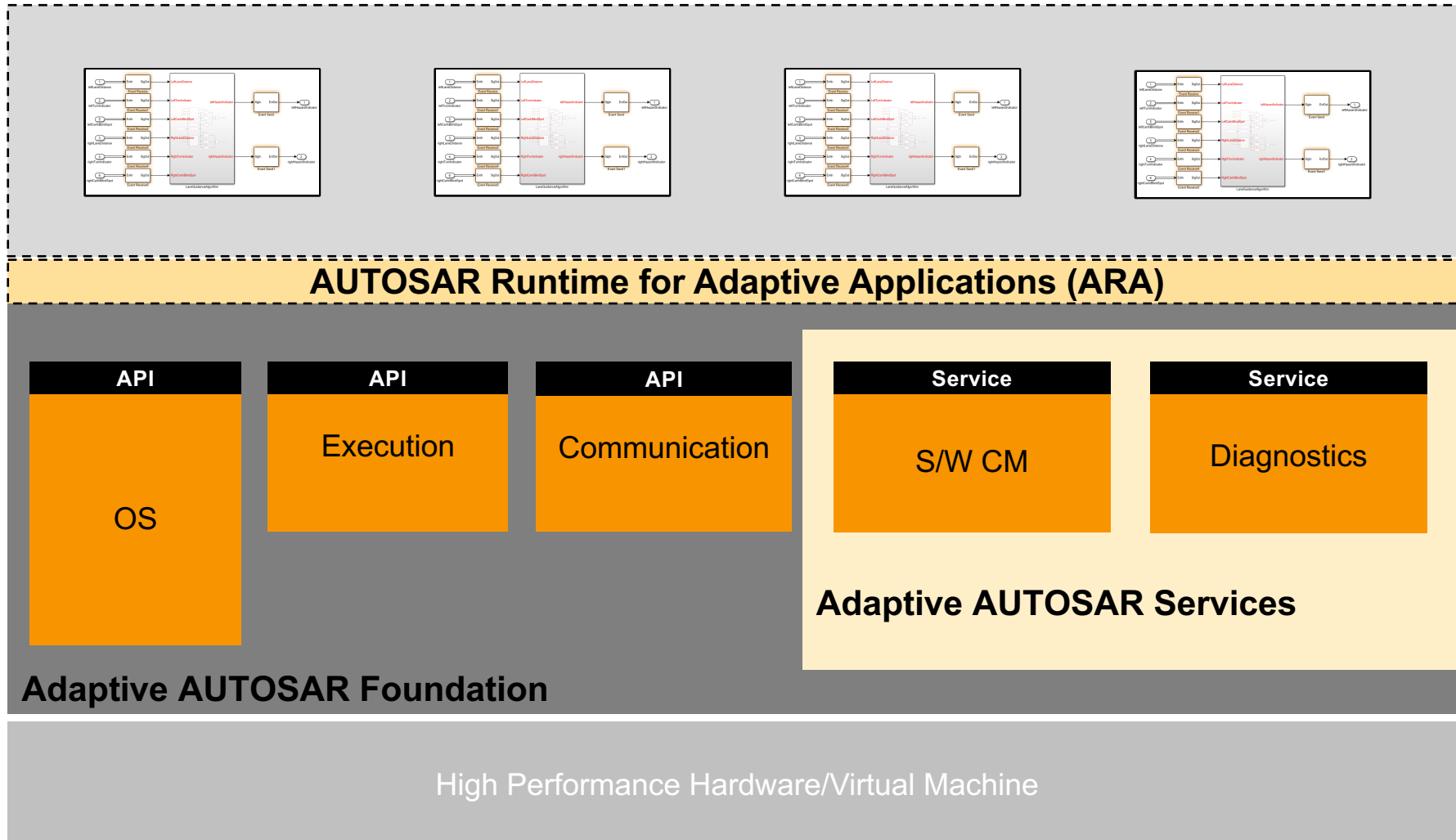
- Software-defined vehicles and new architectures (SOA)
- **New Simulink Semantics**
 - Generic SOA
 - SOA based AUTOSAR Adaptive applications
- Conclusions and key takeaways

AUTOSAR Adaptive

AUTOSAR Adaptive Platform implements the AUTOSAR Runtime for Adaptive Applications (ARA) for automotive industry.

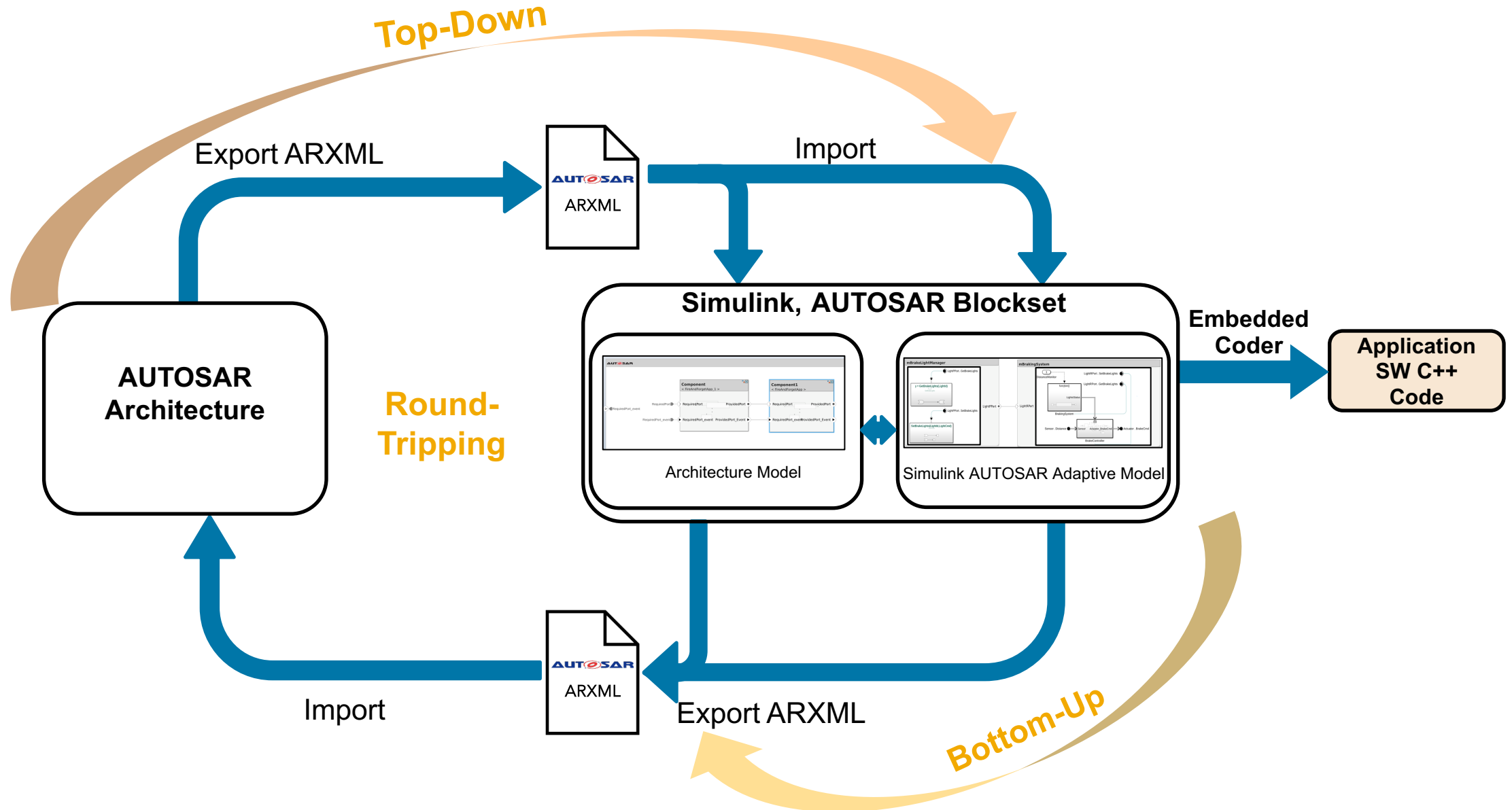


Simulink, AUTOSAR Blockset and Embedded Coder for Adaptive Platform



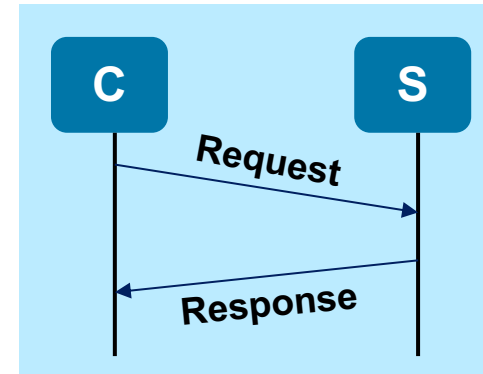
Modeling & C++ Code Generation

AUTOSAR Workflows - Importing and Exporting AUTOSAR Descriptions for Adaptive applications

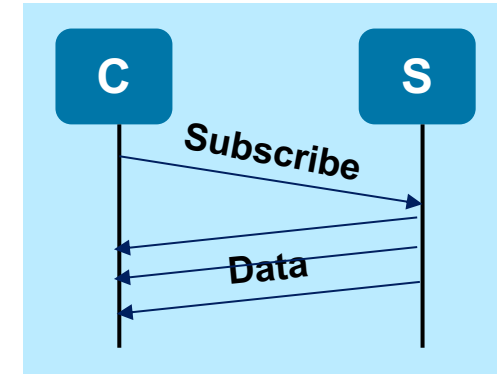


Service-oriented communication used in Adaptive applications

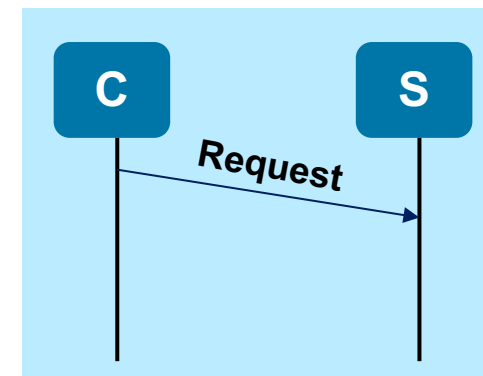
- Service Interface can contain
 - Methods
 - Events
 - Fields



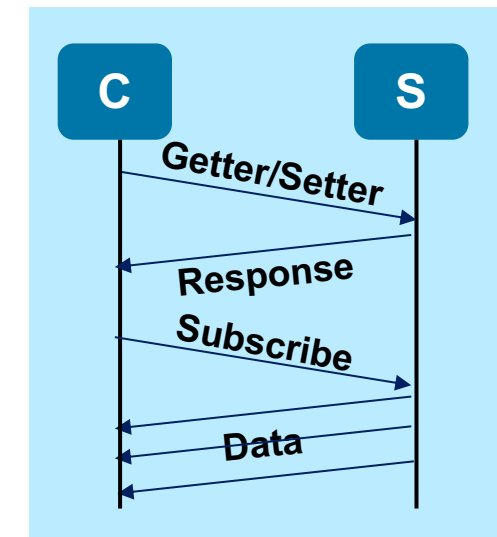
**Method
Request/Response**



Event

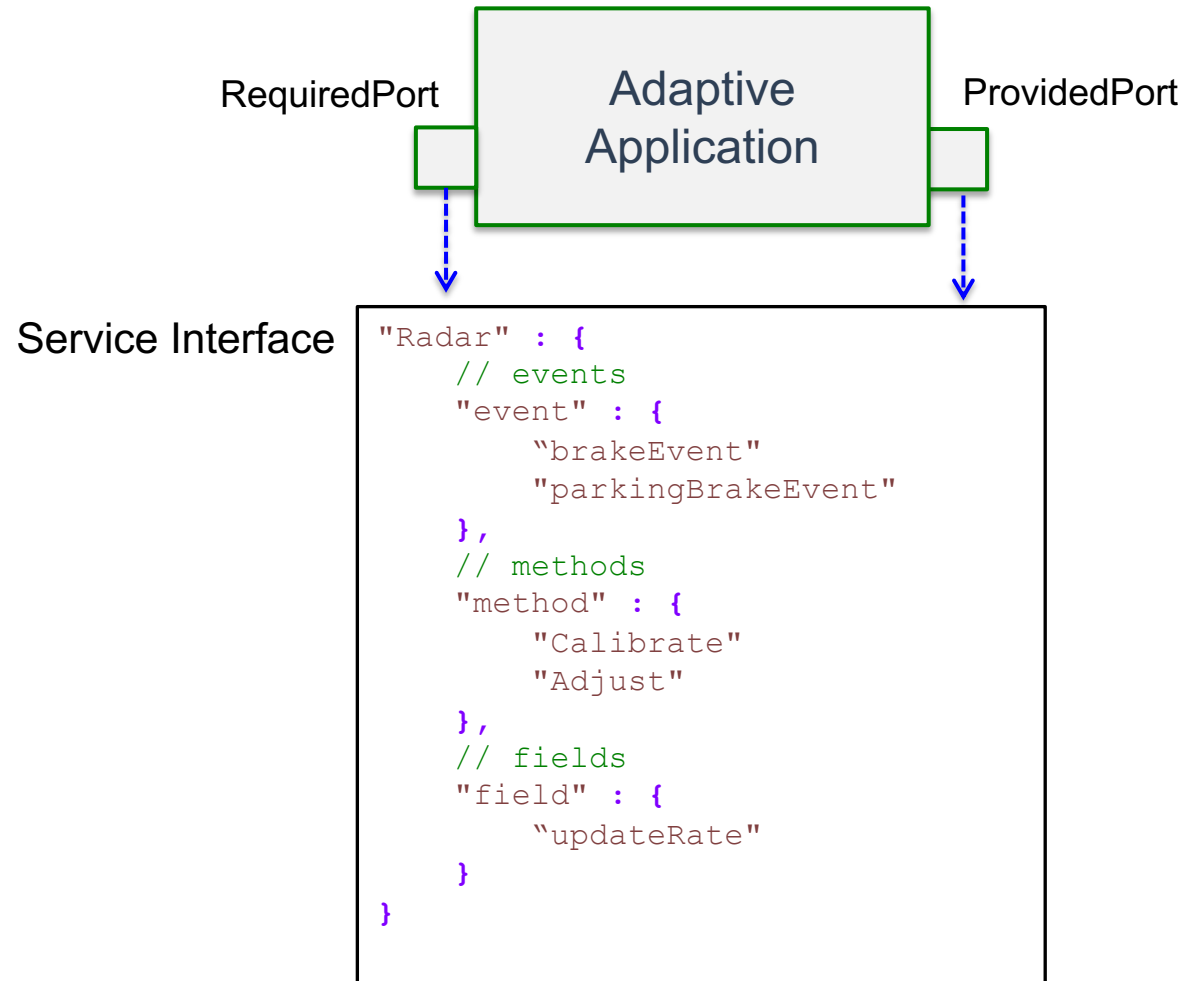


**Method
Fire/Forget**



Field

Adaptive SW Architecture Concepts



Mapping AUTOSAR Adaptive Concepts to Simulink

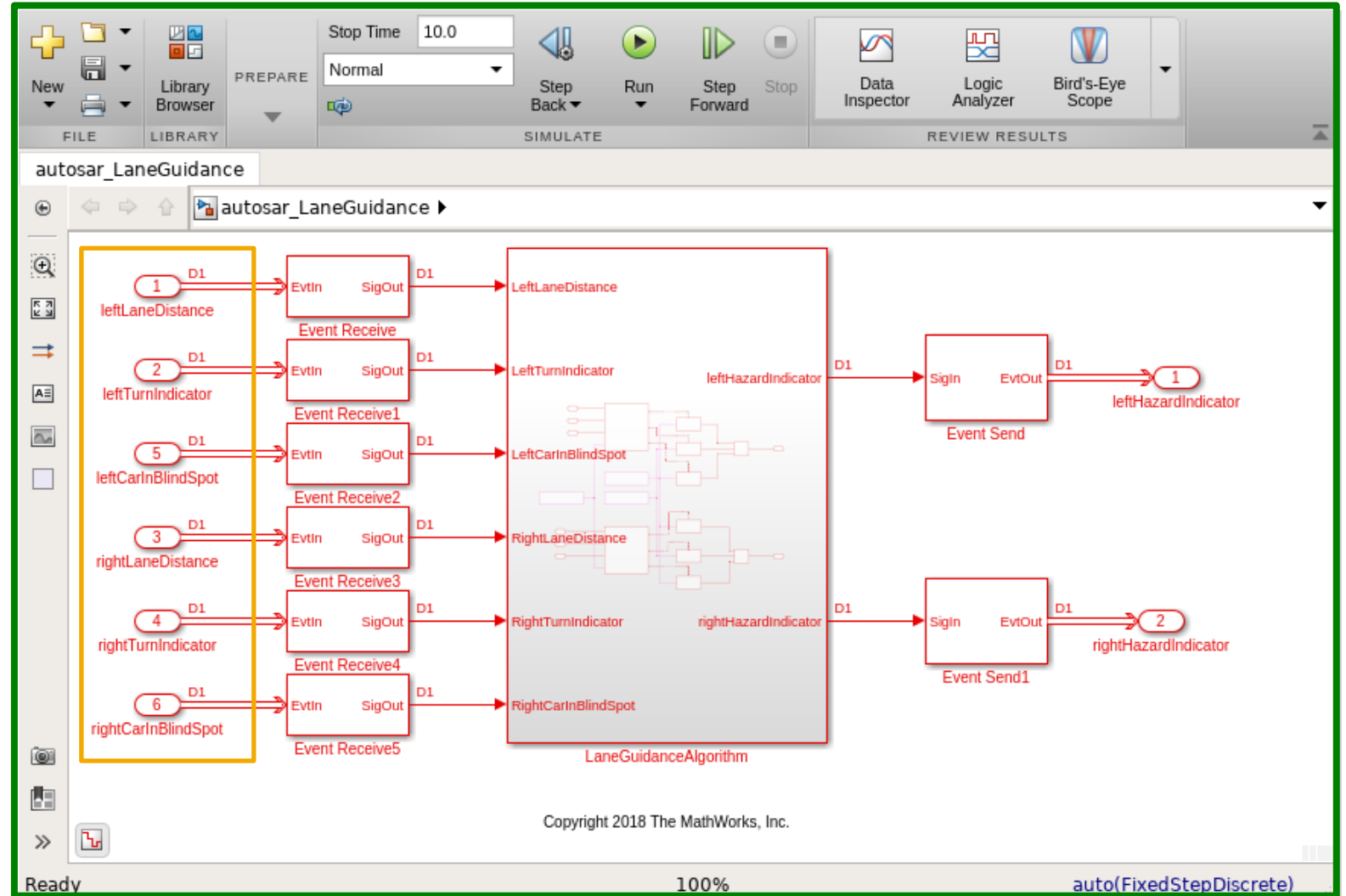


Adaptive Application

RequiredPort

```

"Radar" : {
  // events
  "event" : {
    "leftLaneDistance"
    "leftTurnIndicator"
    "leftCarInBlindSpot"
    "rightLandDistance"
    "rightTurnIndicator"
    "rightCarInBlindSpot"
  },
  // methods
  "method" : {
    "Calibrate"
    "Adjust"
  },
  // fields
  "field" : {
    "updateRate"
  }
}
    
```



Mapping AUTOSAR Adaptive Concepts to Simulink

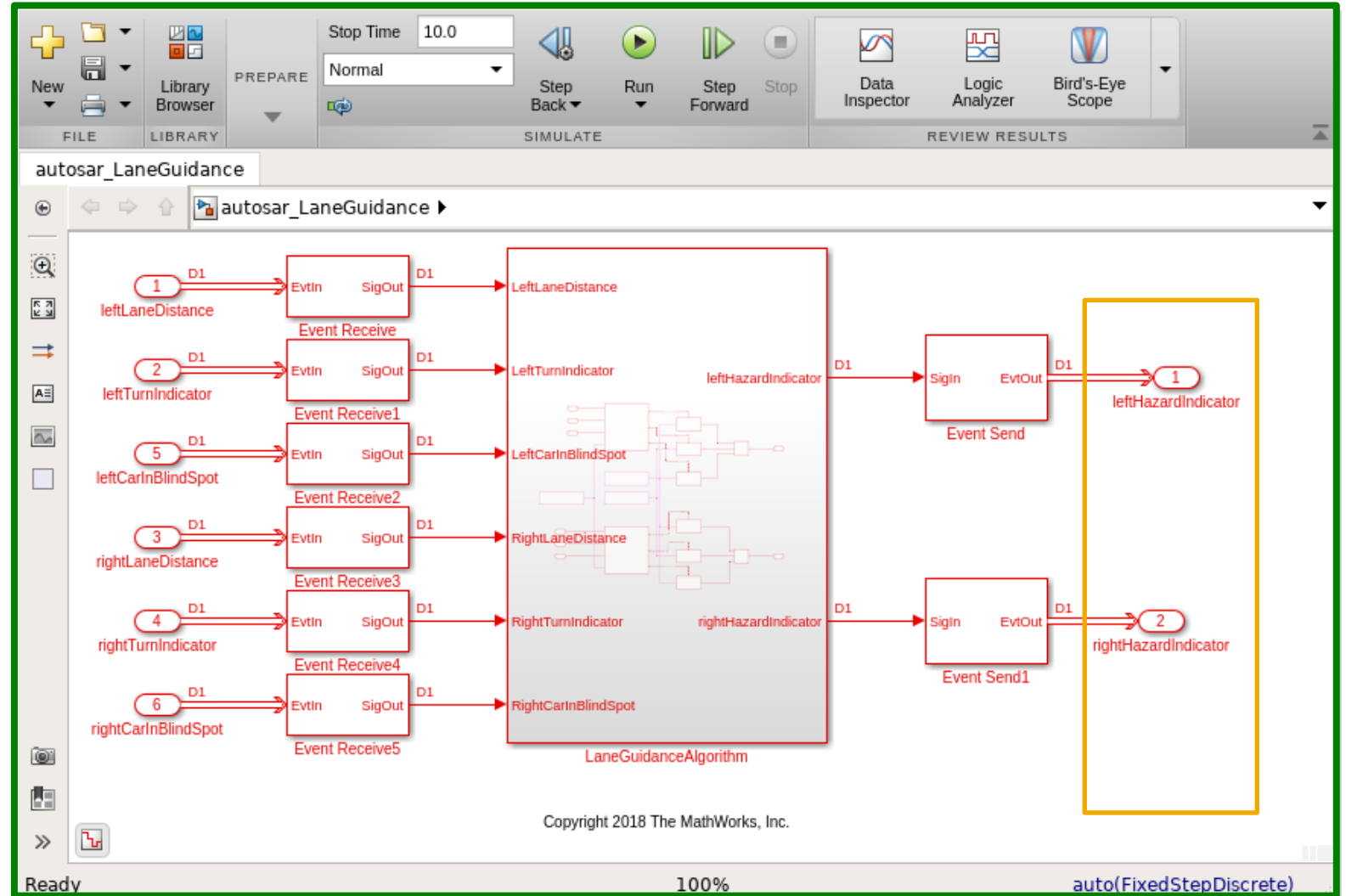


Adaptive Application

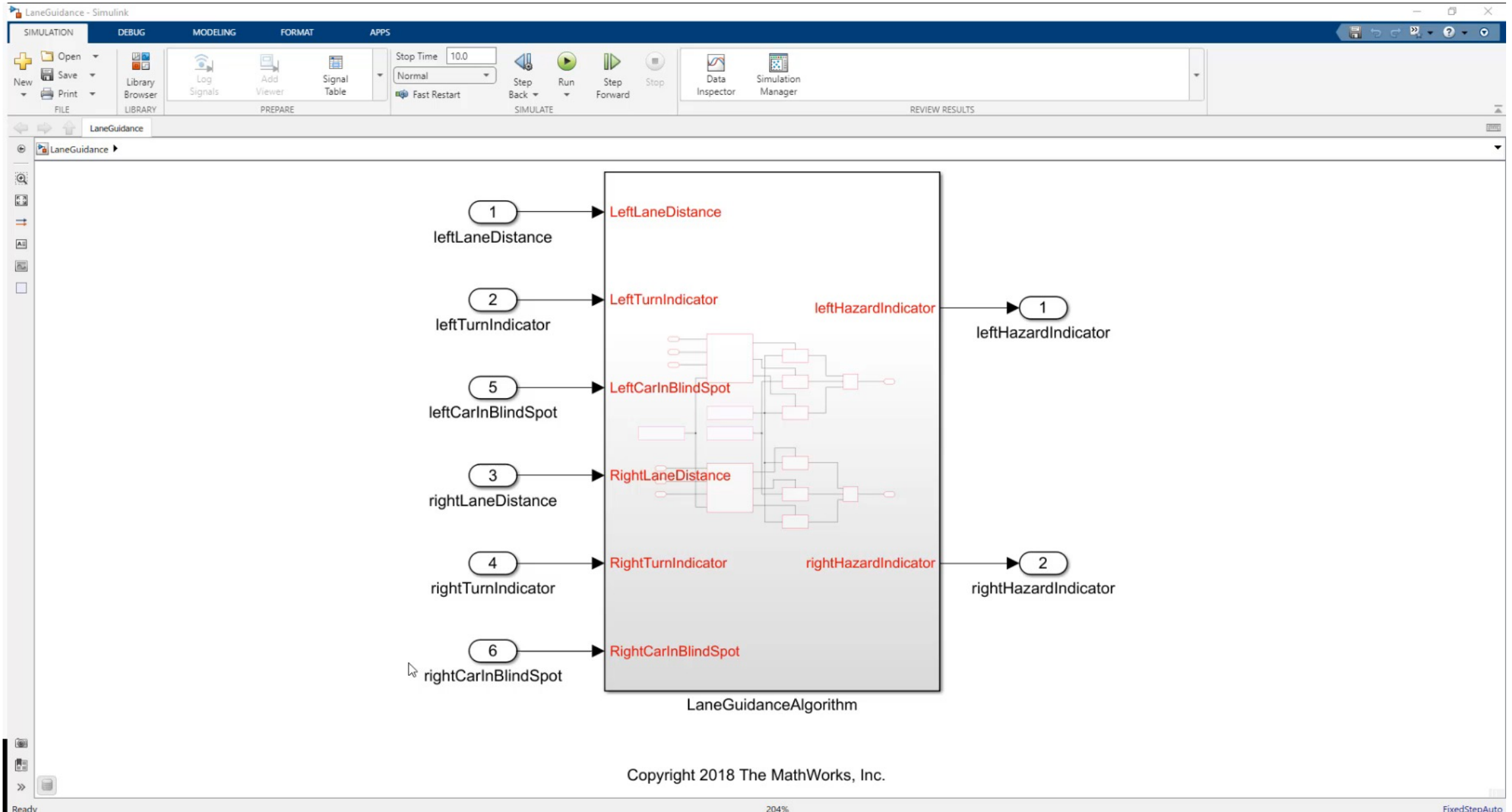
ProvidedPort

```

"Radar" : {
  // events
  "event" : {
    "leftHazardIndicator"
    "rightHazardIndicator"
  },
  // methods
  "method" : {
    "Calibrate"
    "Adjust"
  },
  // fields
  "field" : {
    "updateRate"
  }
}
    
```



Configure for an AUTOSAR Adaptive application

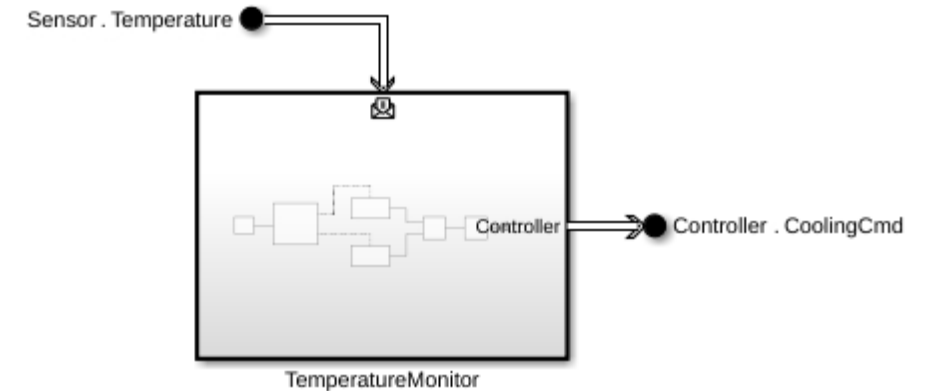


Model event-triggered execution

- Event-triggered execution
 - Application logic only runs when an event arrives
 - Saves computing resources
 - Maps well to applications where events are sporadic



Events trigger the receiver execution



```
void TempController::TemperatureMonitor()
{
    Sensor->Temperature.GetNewSamples(&temperatureValue);
    rtb_Merge = (temperatureValue > 70.0);
    Controller->CoolingCmd.Send(static_cast<real_T>(rtb_Merge));
}

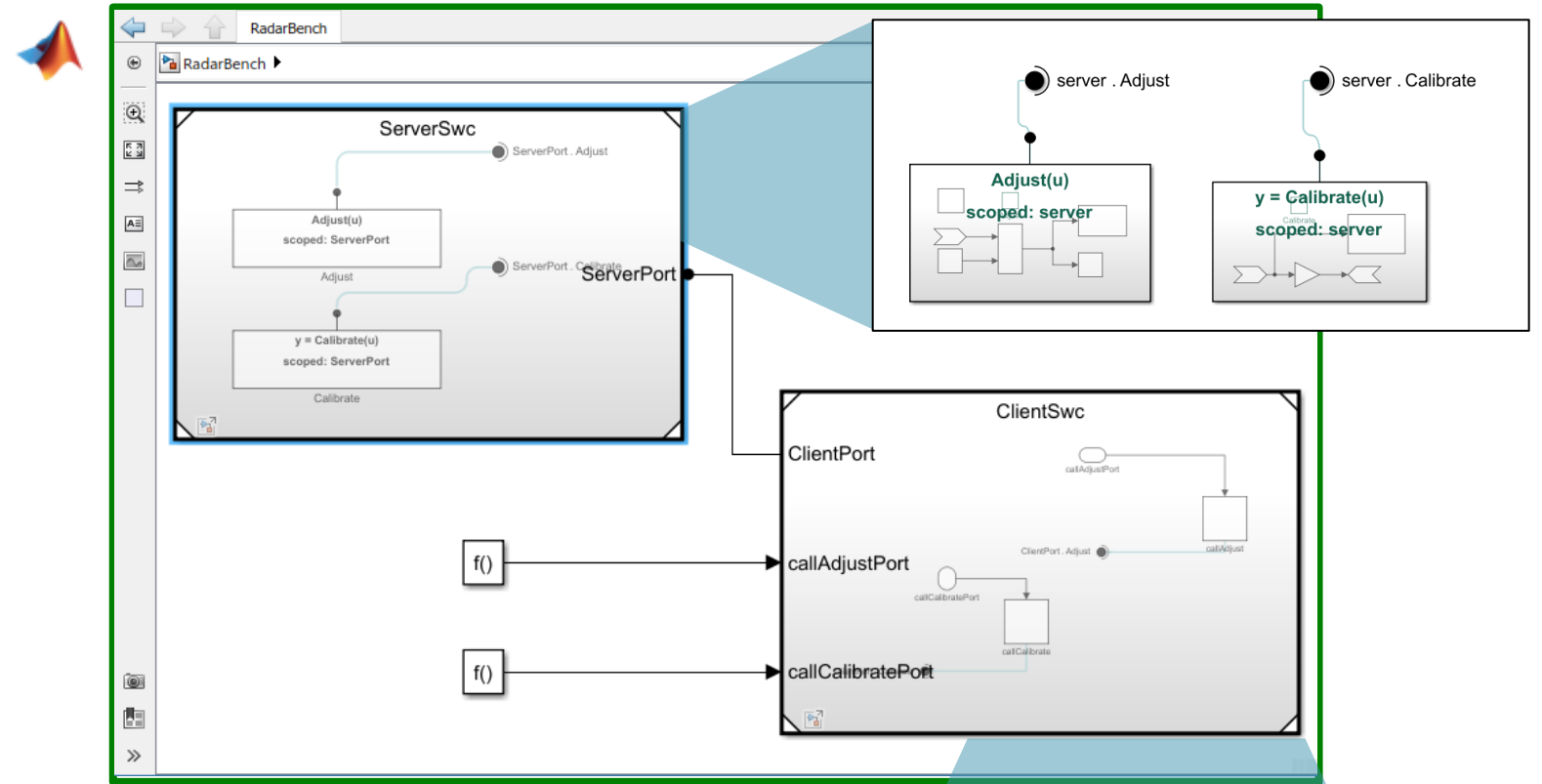
void TempController::initialize()
{
    Controller = initialize_PPort();
    Sensor = initialize_RPort();
    Sensor->Temperature.Subscribe(1U);
    Sensor->Temperature.SetReceiveHandler(
        &TempController::TemperatureMonitor, this);
}
```

Modelling an AUTOSAR Adaptive application in Simulink



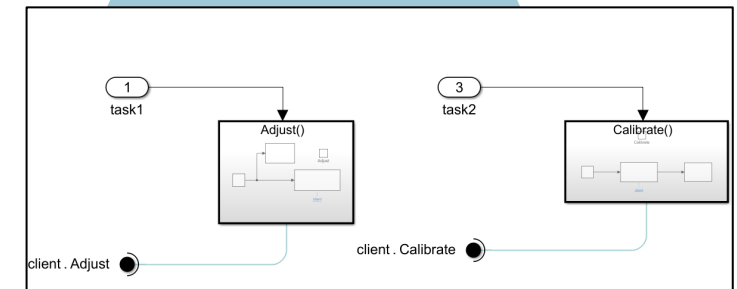
```

"Hazard" : {
  // events
  "event" : { {} },
  // methods
  "method" : {
    "calibrate"
    "adjust"
  },
  // fields
  "field" : { {} }
}
  
```



Modelling Methods

- Blocking Request-Response
- Fire-Forget methods



Model asynchronous/non-blocking methods

- Simulink supports modelling of methods
 - Non-blocking Request-Response methods

```

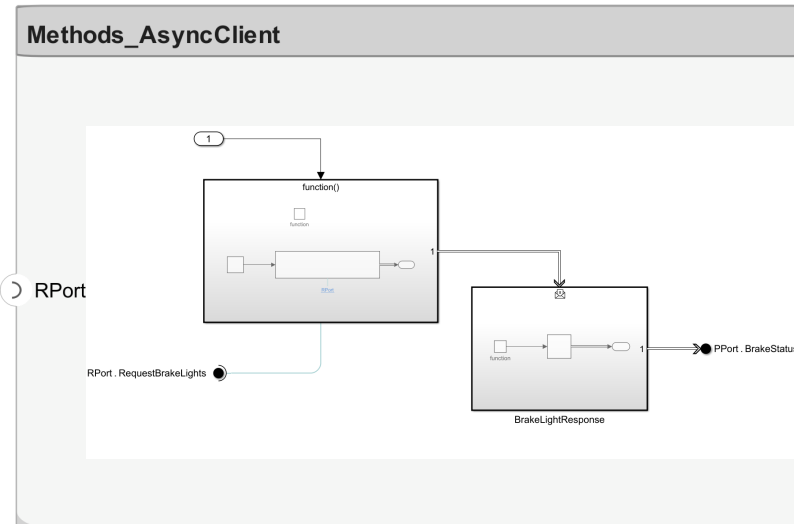
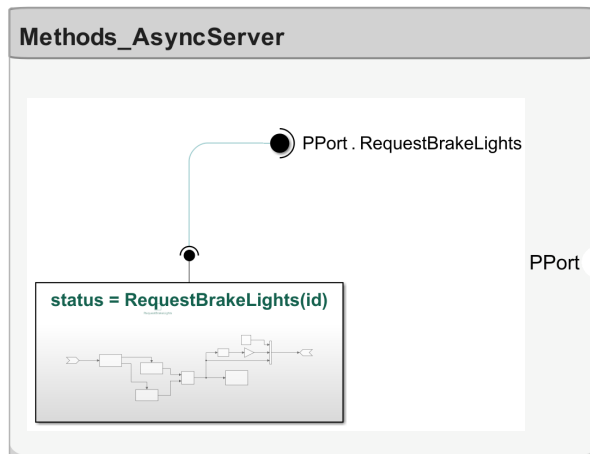
void mAsyncCallerStructOutput::RequestBrakeLightsCallback(ara::core::Future<
proxy::methods::RequestBrakeLights::Output> fObj)
{
    proxy::methods::RequestBrakeLights::Output callOutput;
    std::shared_ptr<ara::core::Result<proxy::methods::RequestBrakeLights::Output>>
        RequestBrakeLightsResultPtr;

    // Retrieve result on method RequestBrakeLights's completion
    RequestBrakeLightsResultPtr = std::make_shared< ara::core::Result<proxy::
        methods::RequestBrakeLights::Output> >(fObj.GetResult());

    // Check if method RequestBrakeLights completed successfully and returned valid results
    if (RequestBrakeLightsResultPtr->HasValue()) {
        // Retrieve return arguments from method RequestBrakeLights's Result container
        callOutput = RequestBrakeLightsResultPtr->Value();
    }

    BrakeLightResponse(&callOutput.status);
}
    
```

Method callback body



Authoring AUTOSAR Adaptive Application in action

The screenshot shows the Simulink environment for authoring an AUTOSAR Adaptive Application. The main workspace displays a block diagram with the following components and connections:

- LaneGuidanceApp** (yellow box) contains a **calibrate** port.
- DetectionsApp** (purple box) contains a **calibrate** port connected to LaneGuidanceApp, and **radarCtrl** and **visionCtrl** ports.
- Radar** (blue box) contains a **radarCtrl** port connected to DetectionsApp.
- Vision** (blue box) contains a **visionCtrl** port connected to DetectionsApp.

The interface includes a top toolbar with tabs for SIMULATION, DEBUG, MODELING, FORMAT, and APPS. A Property Inspector on the right shows details for the 'Main' architecture:

NAME	VALUE
Main	
Name	LaneApplicationSOA
Exported Composition Name	LaneApplicationSOA
Stereotype	Add..
Parameters	
	Select

At the bottom of the window, the status bar shows 'Ready', '77%' zoom, and 'FixedStepDiscrete'.

The Adaptive Standard is developing rapidly, as are our tools!

AUTOSAR Release	MathWorks Release
R18-10	R2019a
R19-03	R2020a
R19-11	R2021a
R20-11	R2022b
R21-11	R2023a

AUTOSAR Adaptive

Generate XML file for schema version:

Maximum SHORT-NAME length:

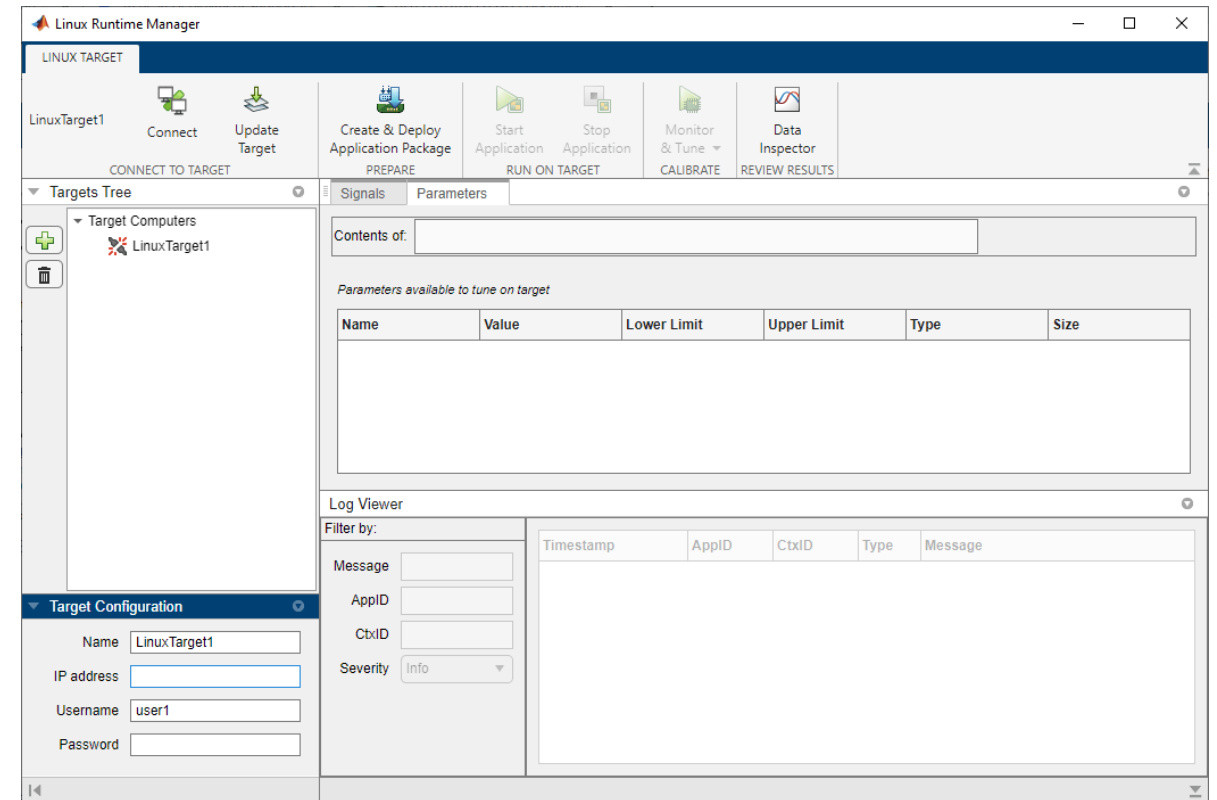
XCP Slave Configuration

Transport layer:

- R18-10 (00046)
- R19-03 (00047)
- R19-11 (00048)
- R20-11 (00049)
- R21-11 (00050)**

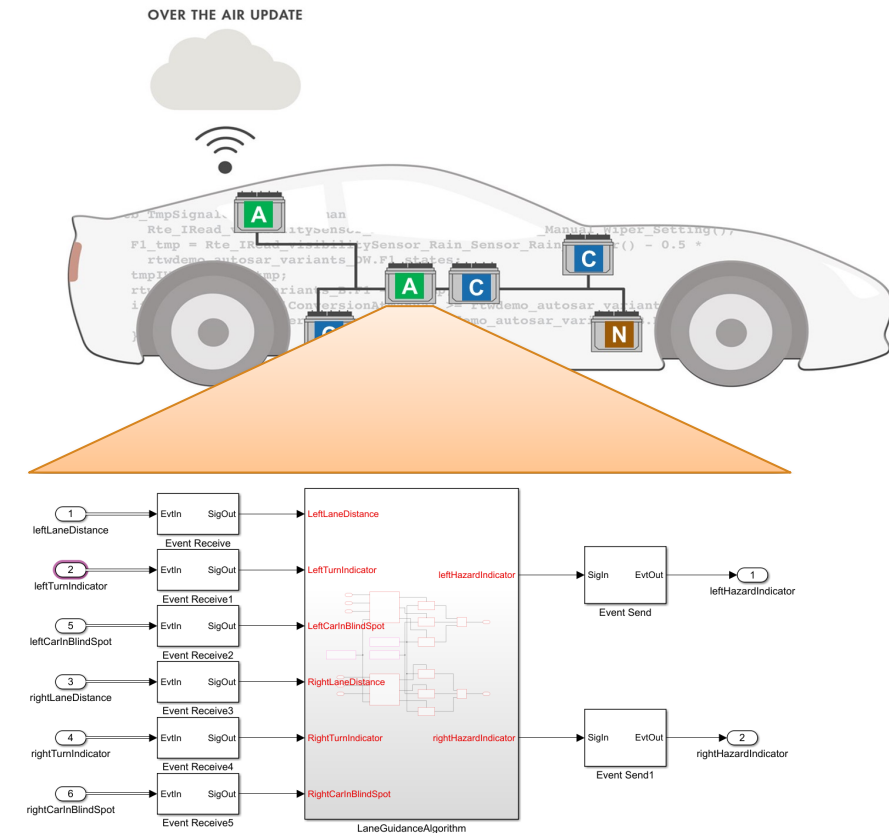
Deploy AUTOSAR Adaptive Architecture Models using Linux Runtime Manager

- Embedded Coder® Support Package for Linux® Applications enables you to deploy the AUTOSAR adaptive architecture models on to target using the Linux Runtime Manager (Embedded Coder) application



Summary

- Design, model, simulate and test AUTOSAR Adaptive Software
 - Author and Model adaptive applications with events and methods communication
 - Import and export adaptive ARXML files
 - AUTOSAR Adaptive Deployment
- Generate optimized AUTOSAR C++ code



Conclusions

Challenges

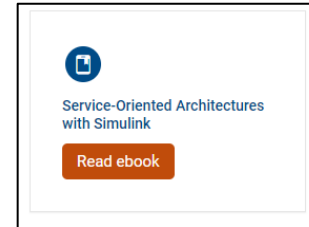
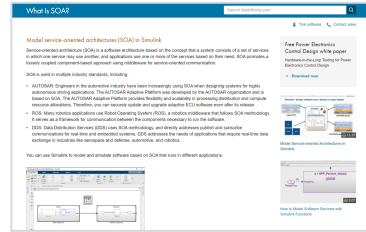
- Development of Adaptive applications require a **change of mindset**
- **Centralize, re-architect** existing applications and partition in processes and services

Solutions

- **Design, simulate and generate** code to deploy service-oriented applications AUTOSAR Adaptive in **Simulink**
- **Reuse your existing expertise and models** to mitigate the risk of migration to Adaptive applications

Call to action

- [SOA Webpage](#)



- [AUTOSAR Blockset](#)



- Technical Paper from Embedded World 2022 - [Develop and Integrate AUTOSAR Classic and Adaptive Applications Based on SOME/IP](#)

- MathWorks presentation at AUTOSAR Open Conference 2023, May 11-12, San Diego, USA

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