Evolution of Simulink Semantics for AUTOSAR Adaptive Applications

Sep 21, 2023 | Troy, Michigan

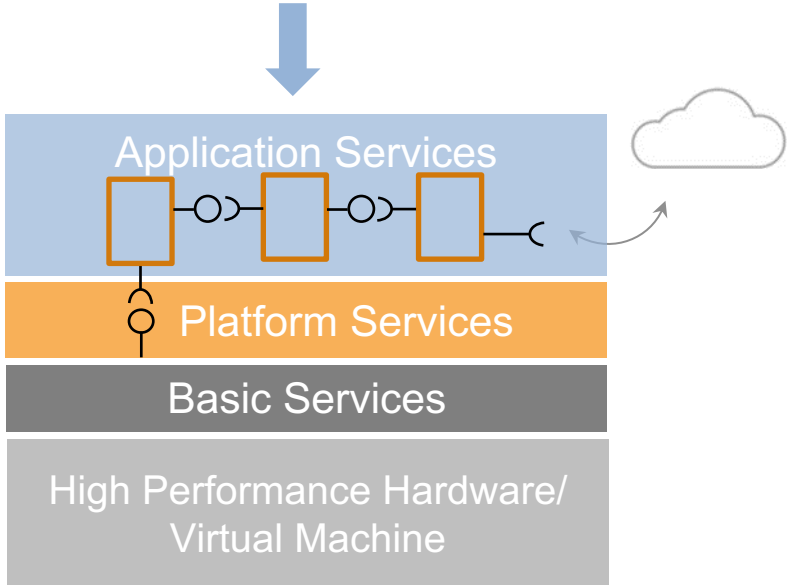
Jeff Harper

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

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

demand

Customer expectations

- Clean and Safe mobility
- Digital Life continuity

Technology & Innovation

- Electrification
- Autonomy
- Connectivity

invest

Business opportunity

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

monetize

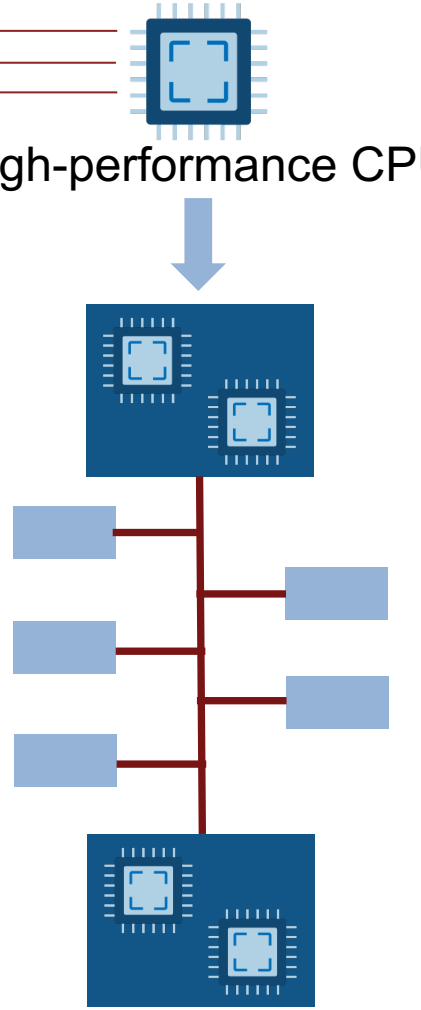
Centralization of computing and SOA

Exponential growth of SW features

100110
001010
010010

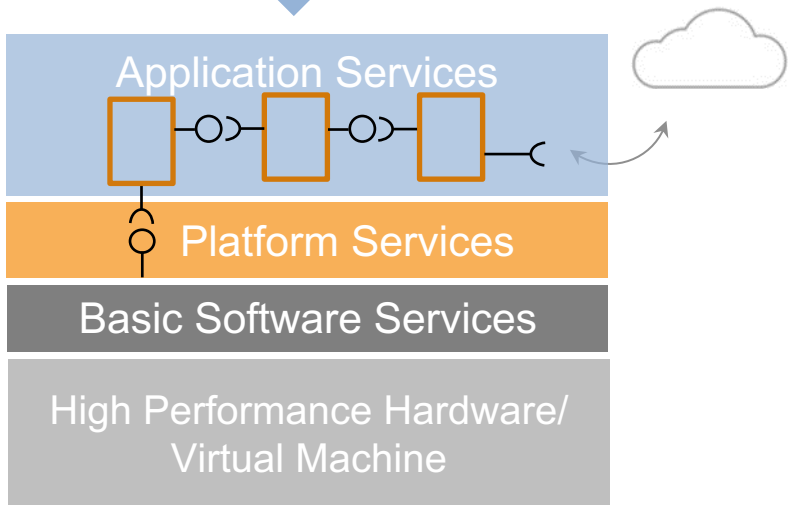
100110
001010
010010

100110
001010
010010



New E/E zonal architectures

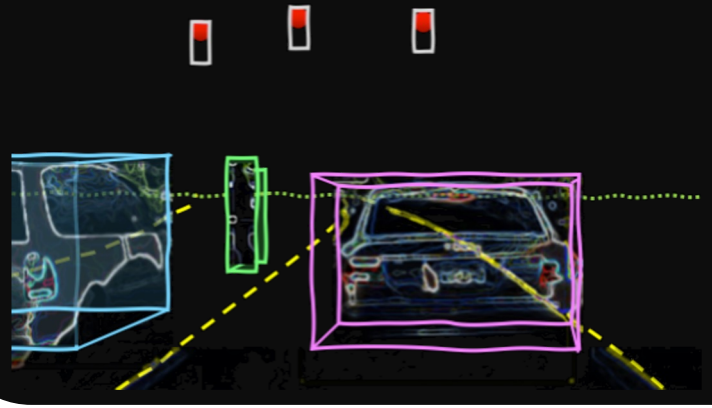
SW updates
- Frequent
- Selective
- Over-the-air



Higher HW abstraction:
Service-oriented architectures

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

Software Defined Vehicle

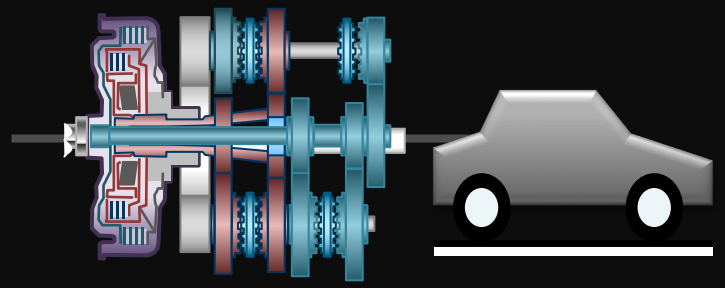


Autonomy
Connectivity
AI

Steering,
Braking

Speed,
Velocity

Traditional Vehicle



Controls
Real-time
CAN

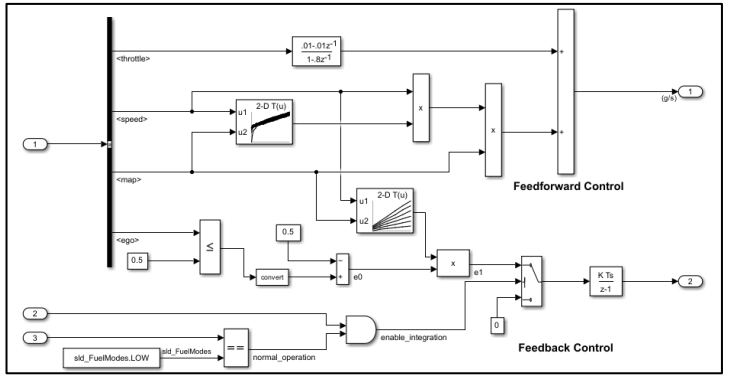
SOA based standards

AUTOSAR Adaptive Platform

ROS

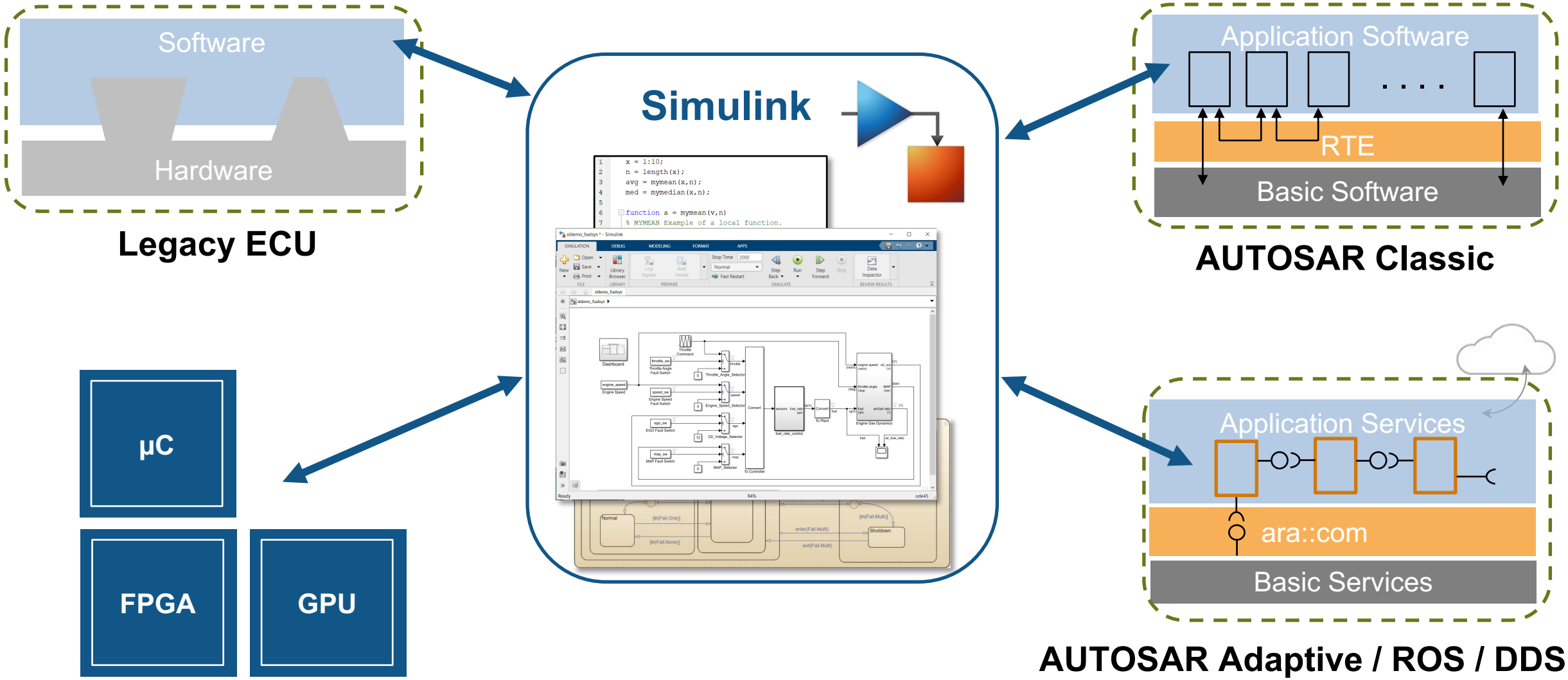
DDS

Model-Based Design



AUTOSAR

Simulink: deploy software to different targets and standards

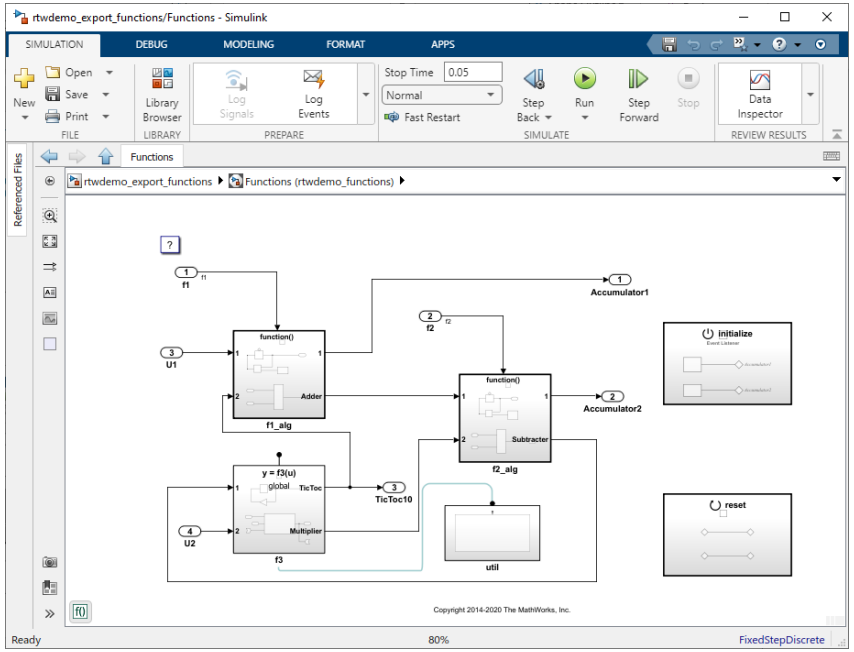


Agenda

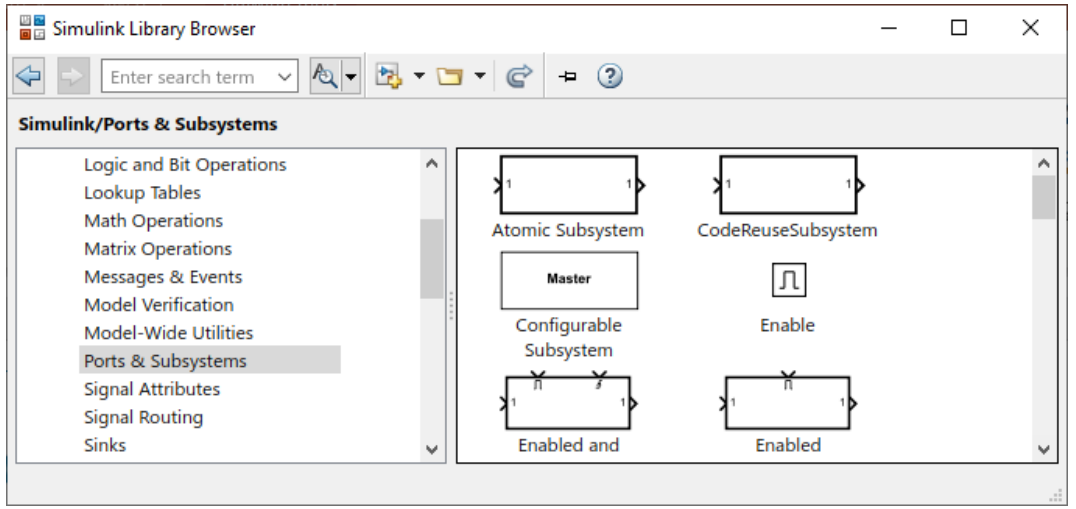
- Software-defined vehicles and new architectures (SOA)
- New Simulink Semantics
  - Generic SOA
  - SOA based AUTOSAR Adaptive applications
- Conclusions and key takeaways

Simulink Supports Exporting Callable Functions Well

- Export Function Modeling style



rtwdemo_export_functions

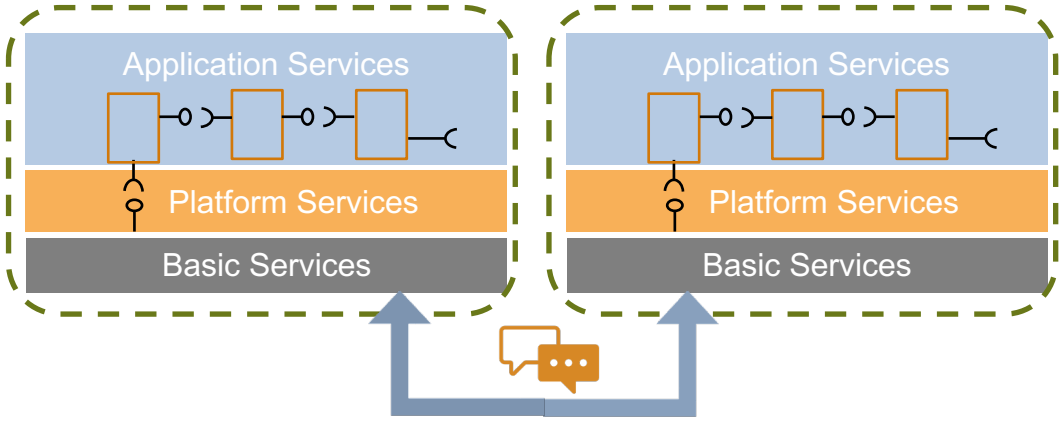


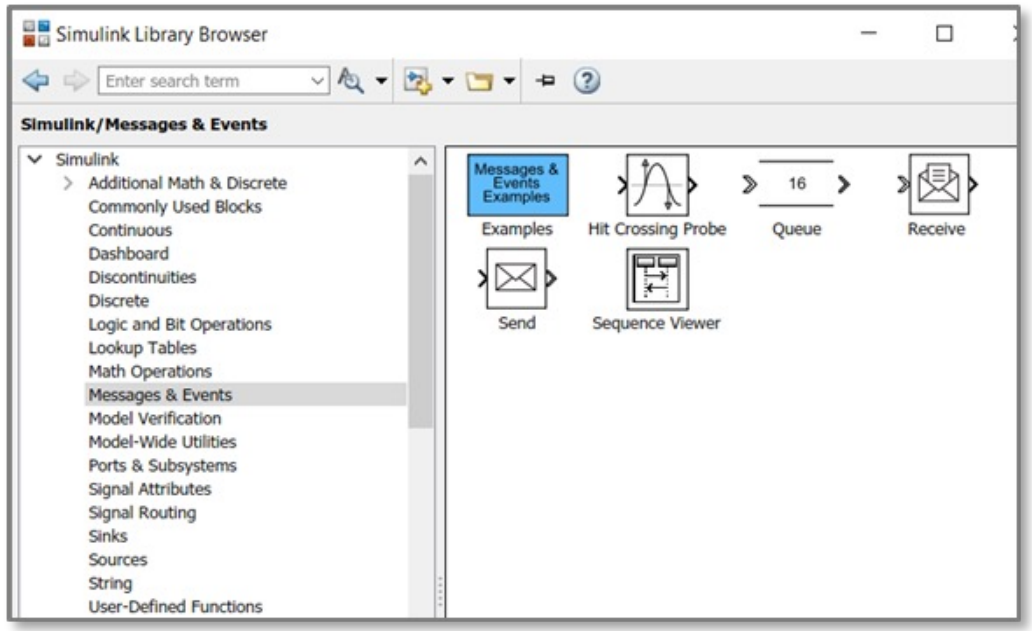
Ports and Subsystems

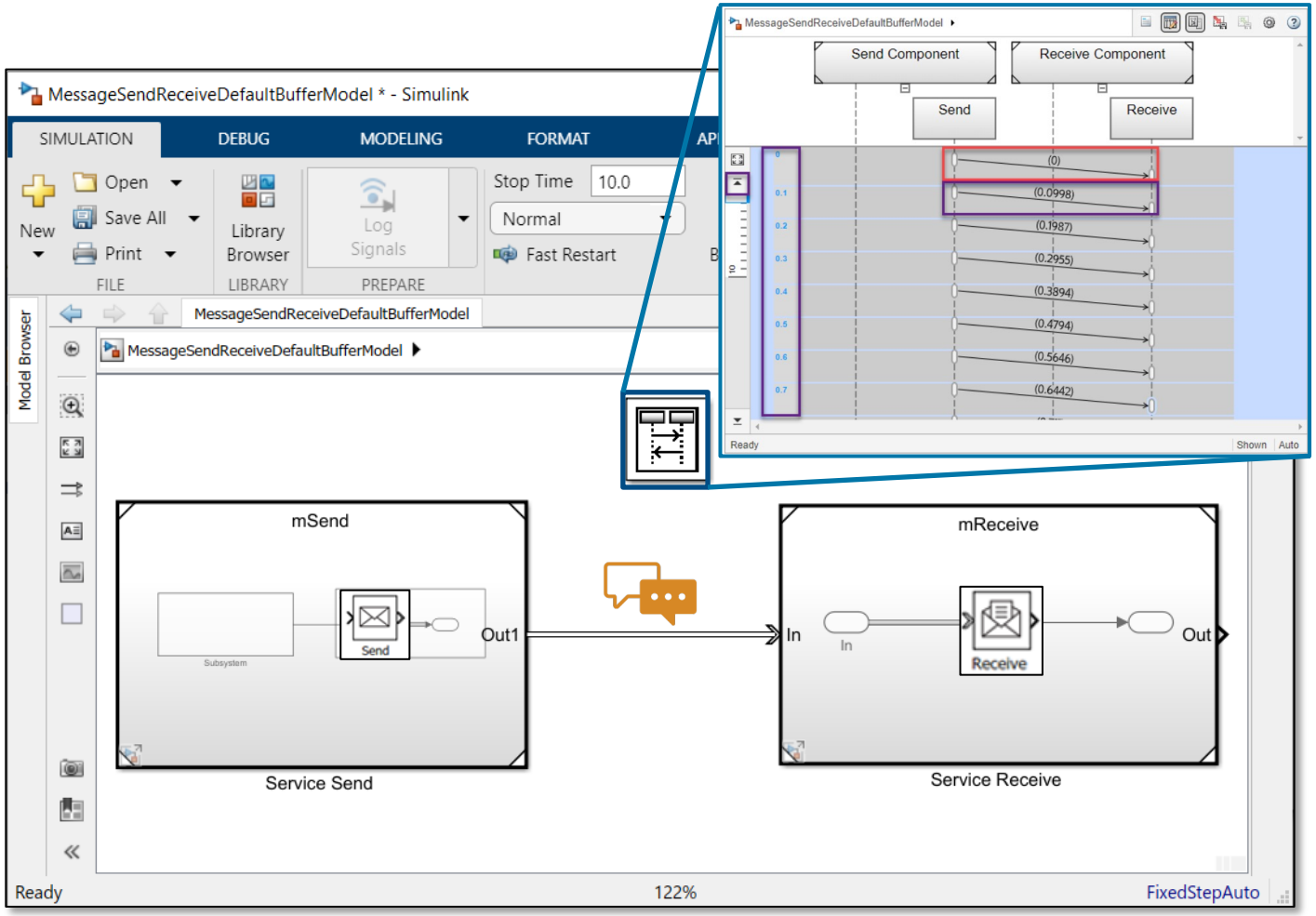
Agenda

- 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



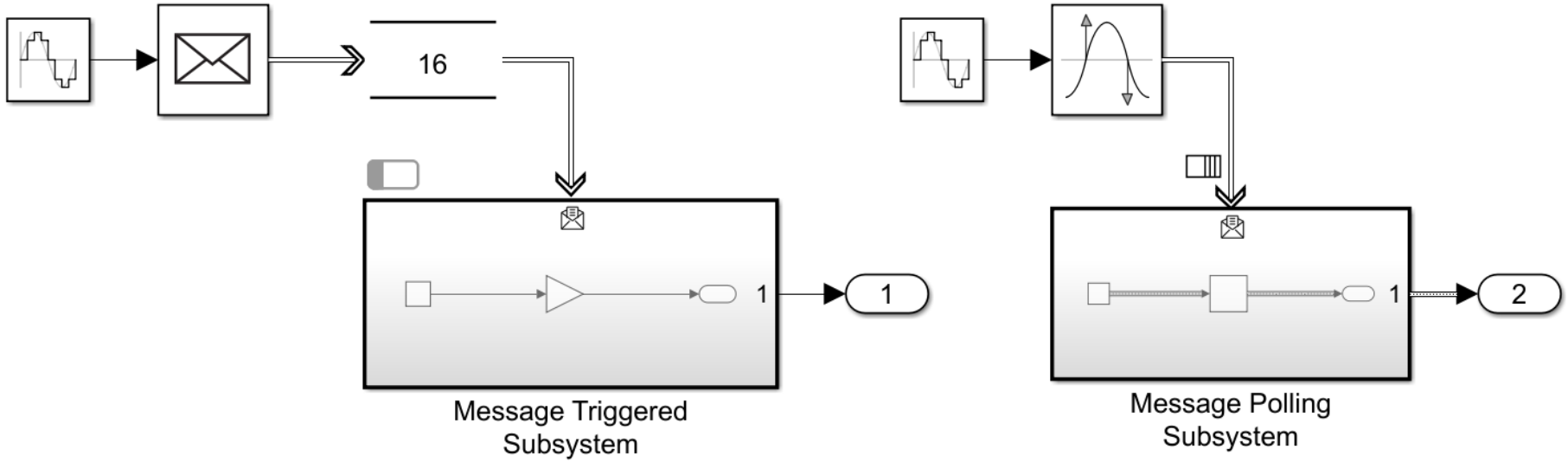




You can model service-oriented communication using messages (Send/Receive)

https://www.mathworks.com/help/simulink/ug/simulink-messages-overview.html

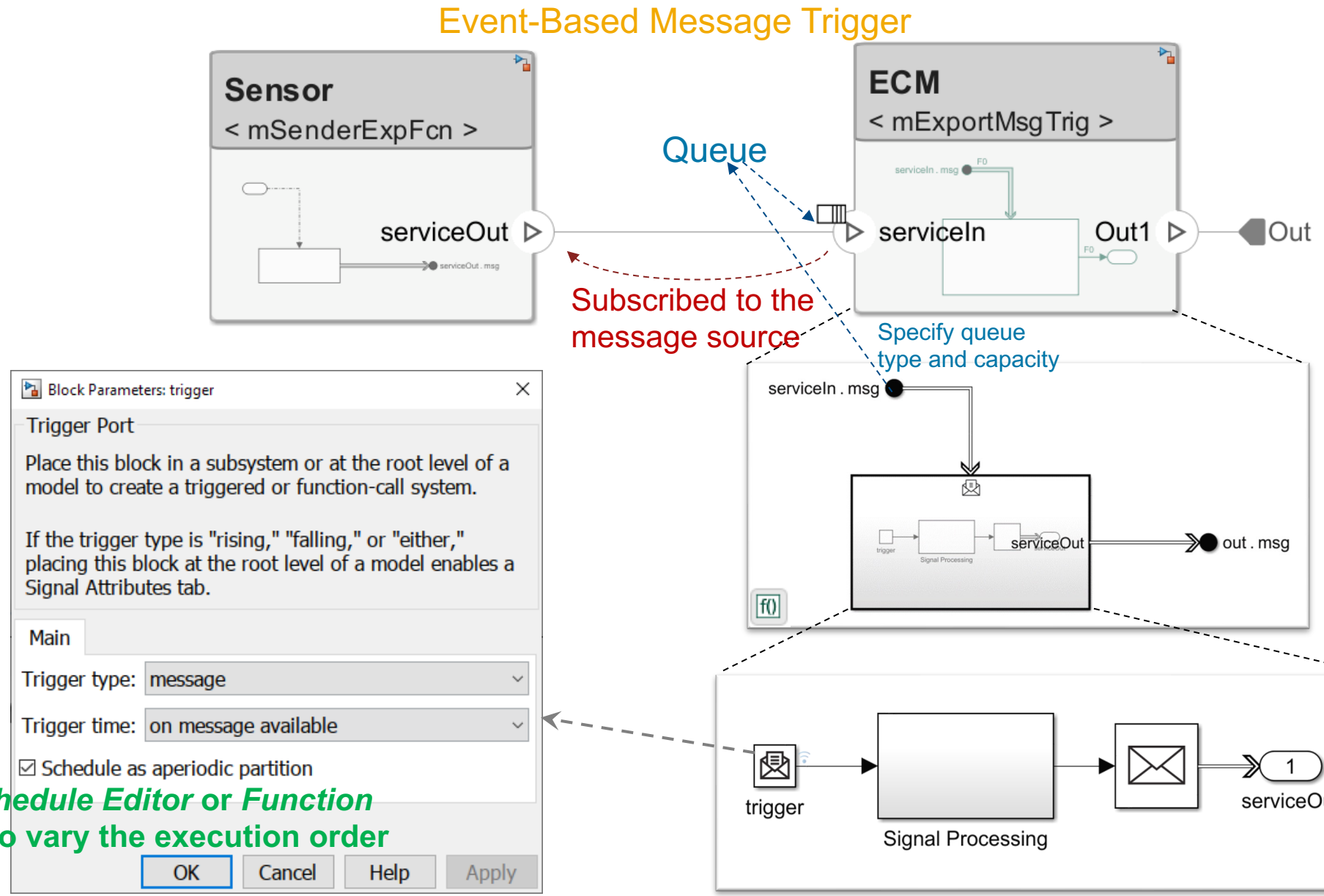
Service-Oriented Behavior Modeling with Message Triggered/Polling Subsystem



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

https://www.mathworks.com/help/simulink/ug/message-triggered-and-message-polling-subsystems.html

Service-Oriented Behavior Modeling with Message Triggered/Polling Subsystem

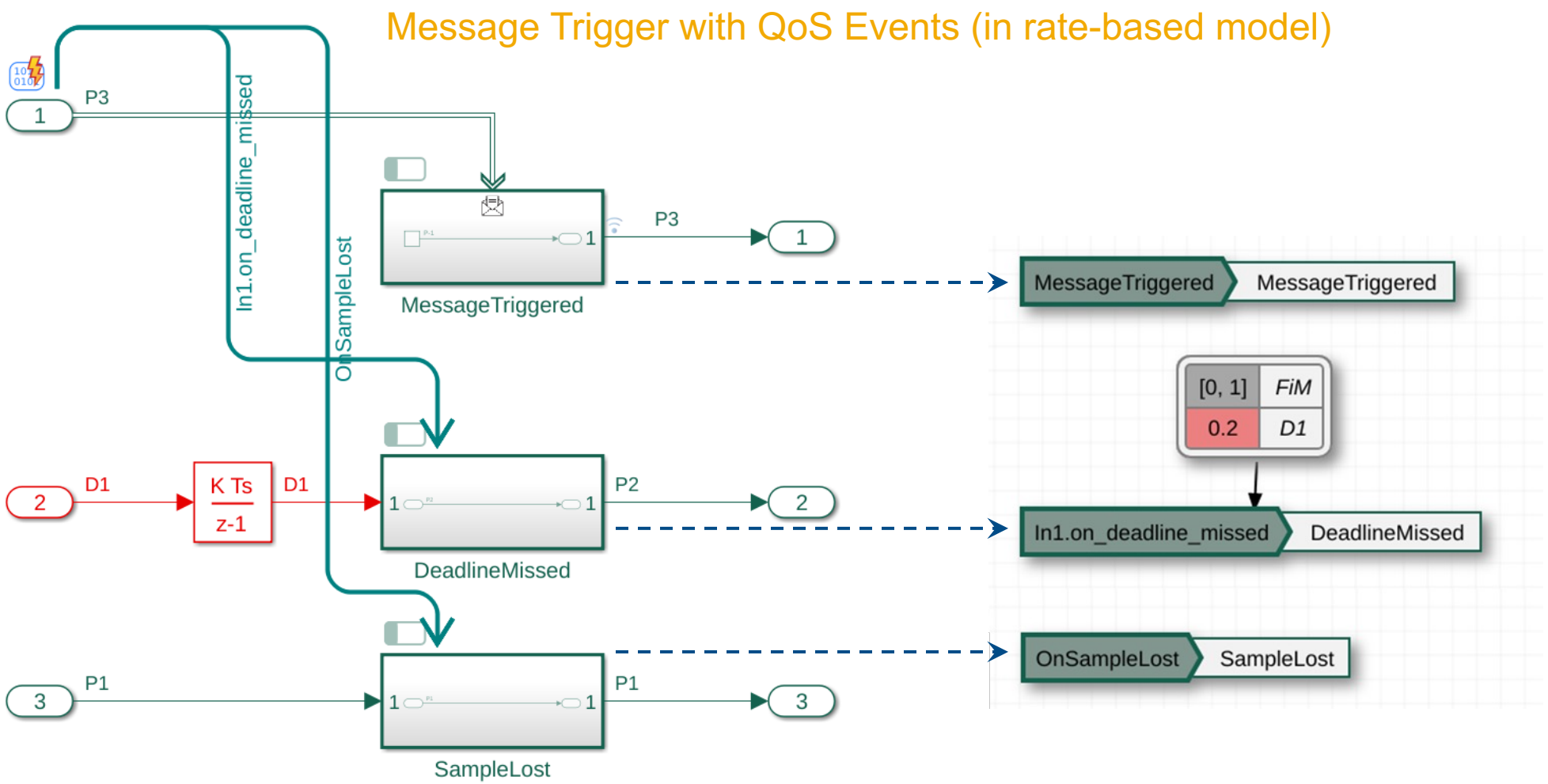


Code gen target supported:

- *AUTOSAR Adaptive*
- *ERT C++*

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

Service-Oriented Behavior Modeling with Message Triggered/Polling Subsystem



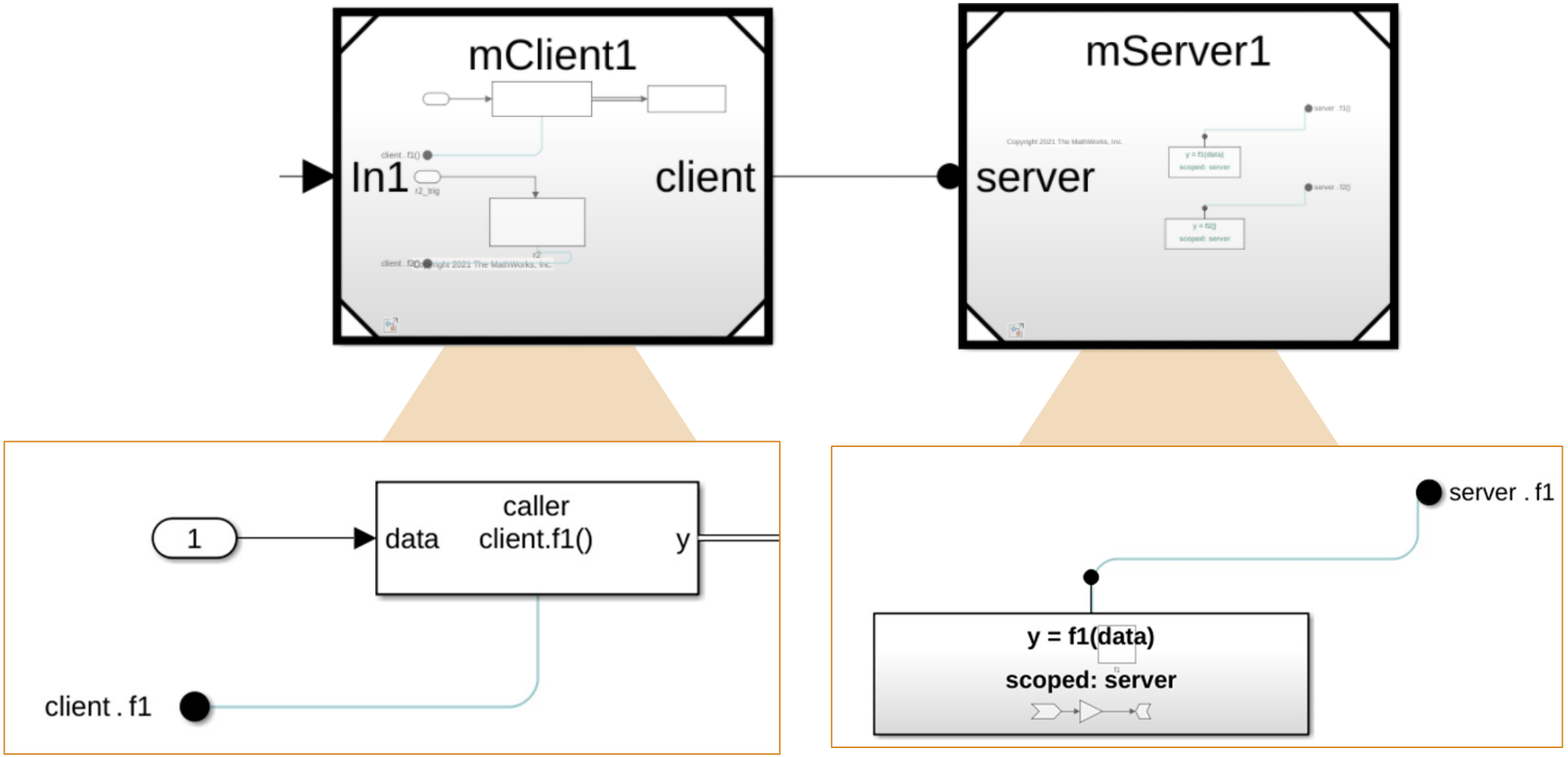
Simulink

Schedule Editor

https://www.mathworks.com/help/simulink/slref/messagetriggeredsubsystem.html

Service-Oriented Behavior Modeling with Service-Based Functions

Function Ports for SOA

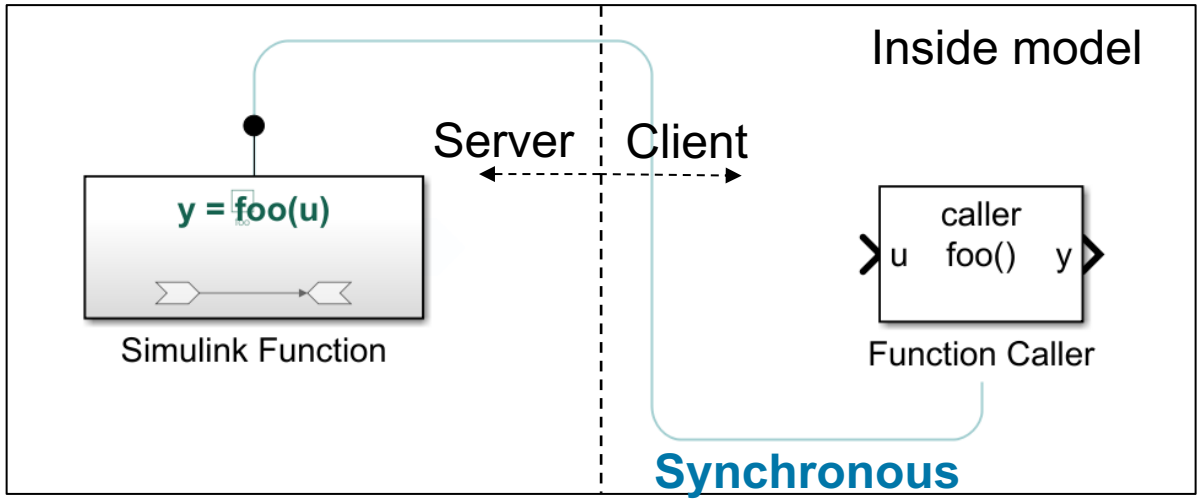


Model client and server components to facilitate data sharing using a functional interface between component models

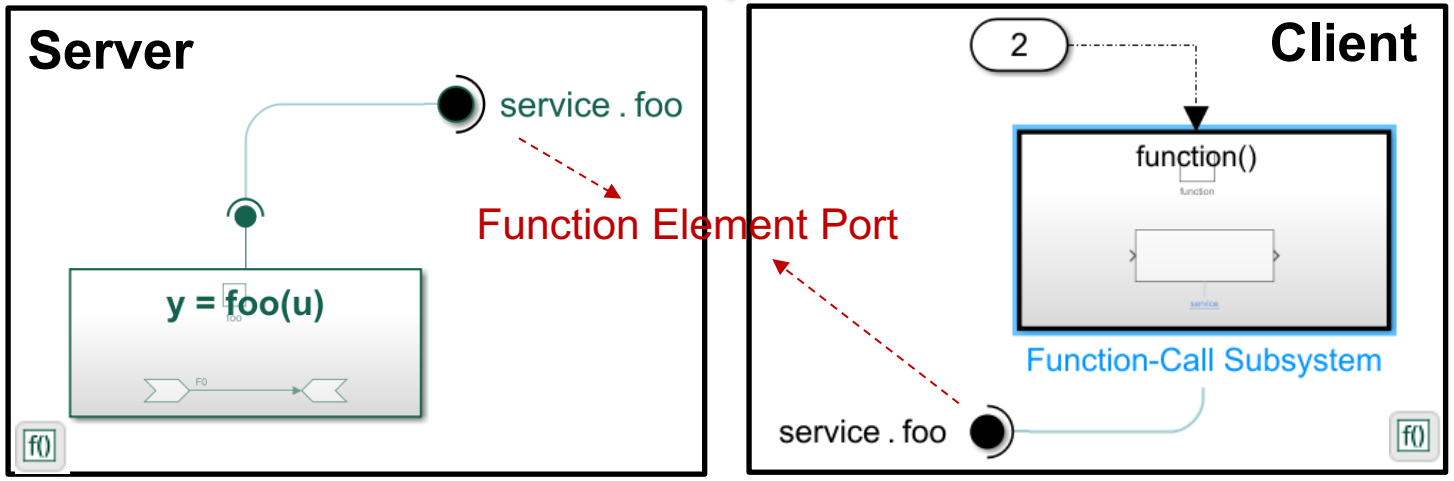
https://www.mathworks.com/help/simulink/ug/call-simulink-functions-in-other-models-using-function-ports.html

Service-Oriented Behavior Modeling with Service-Based Functions

Behavior Model for Services Based on Simulink Function



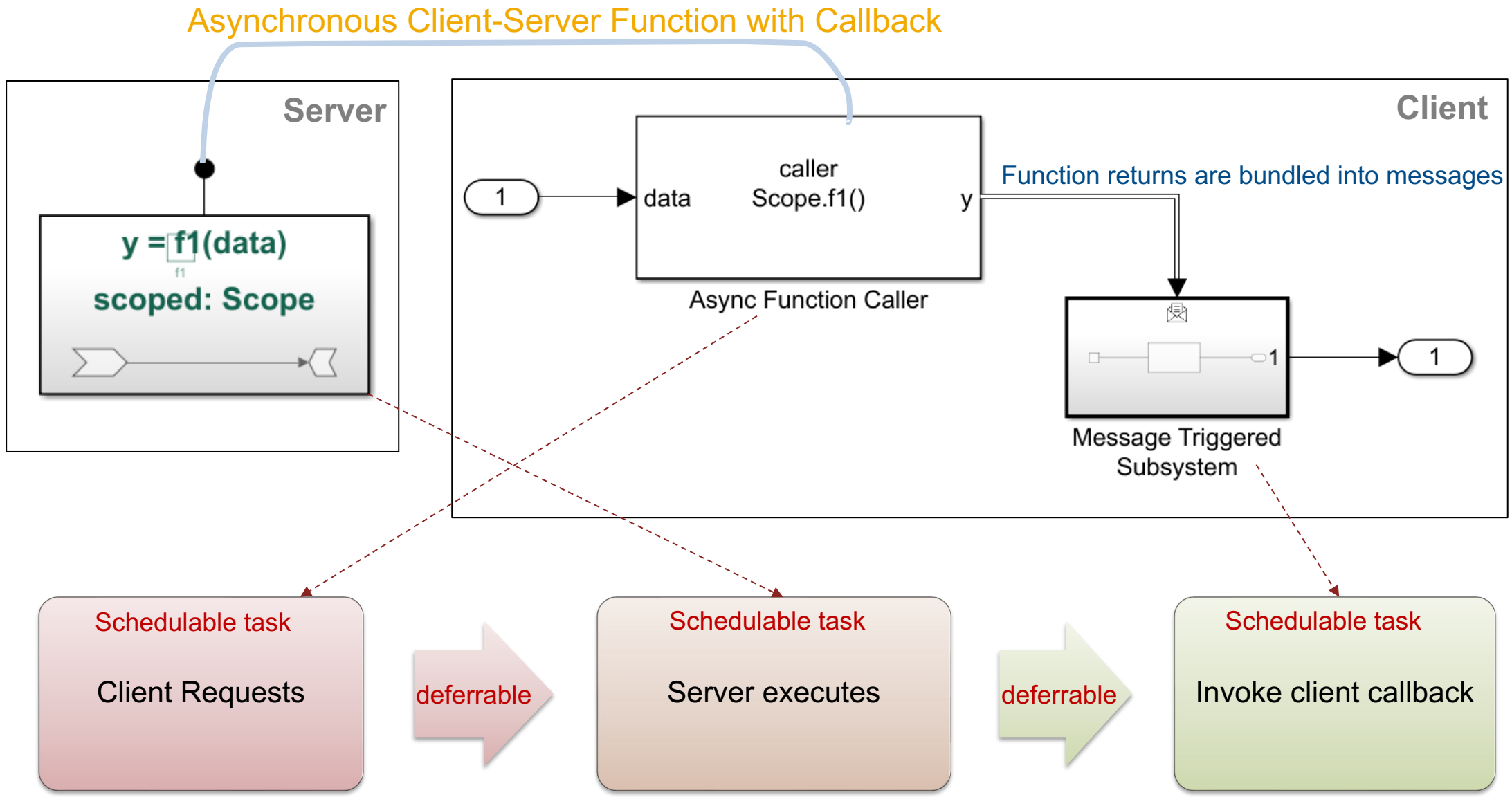
modularization



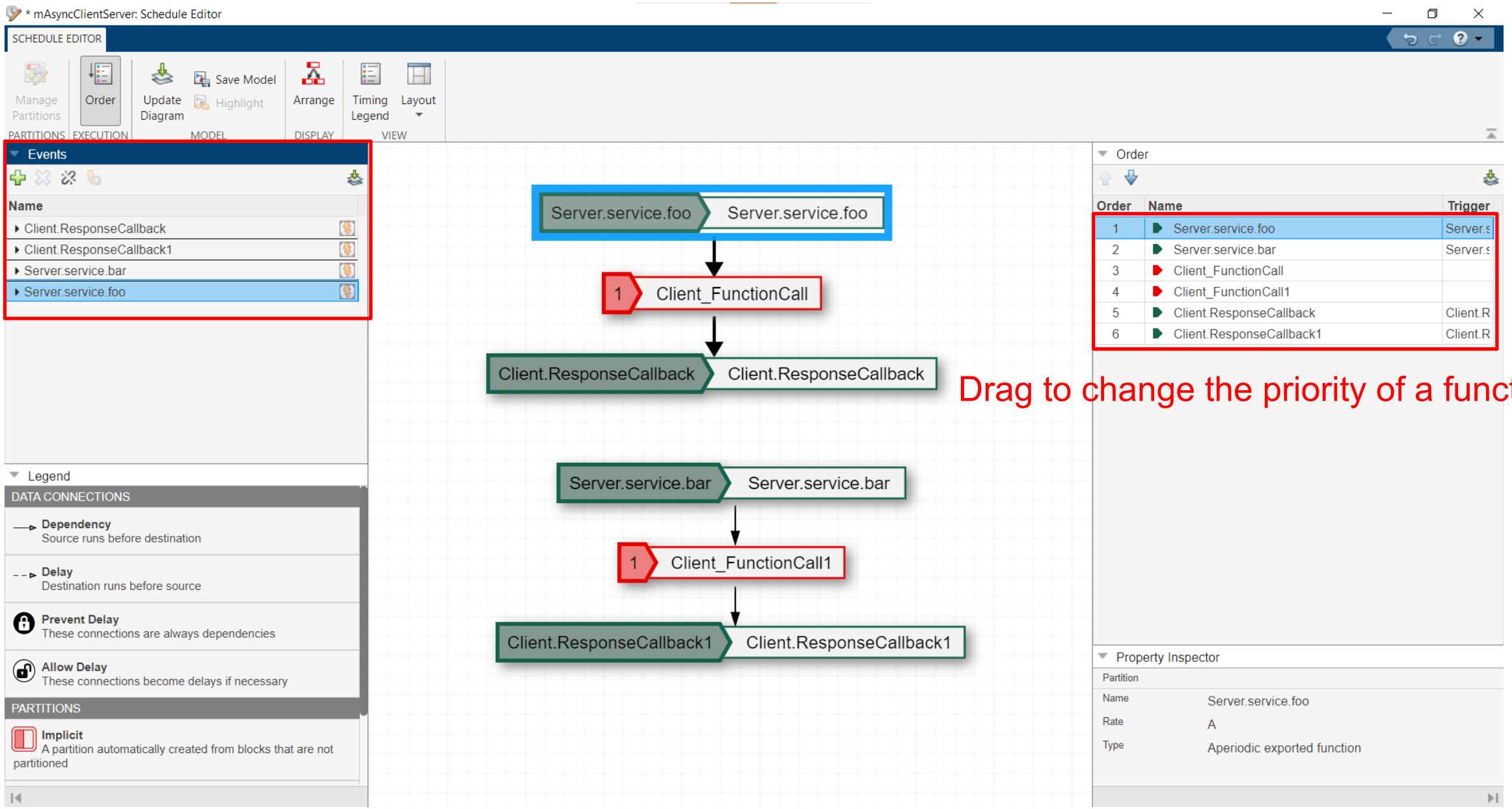
Across model

https://www.mathworks.com/help/simulink/slref/functionelement.html

Service-Oriented Behavior Modeling with Service-Based Functions



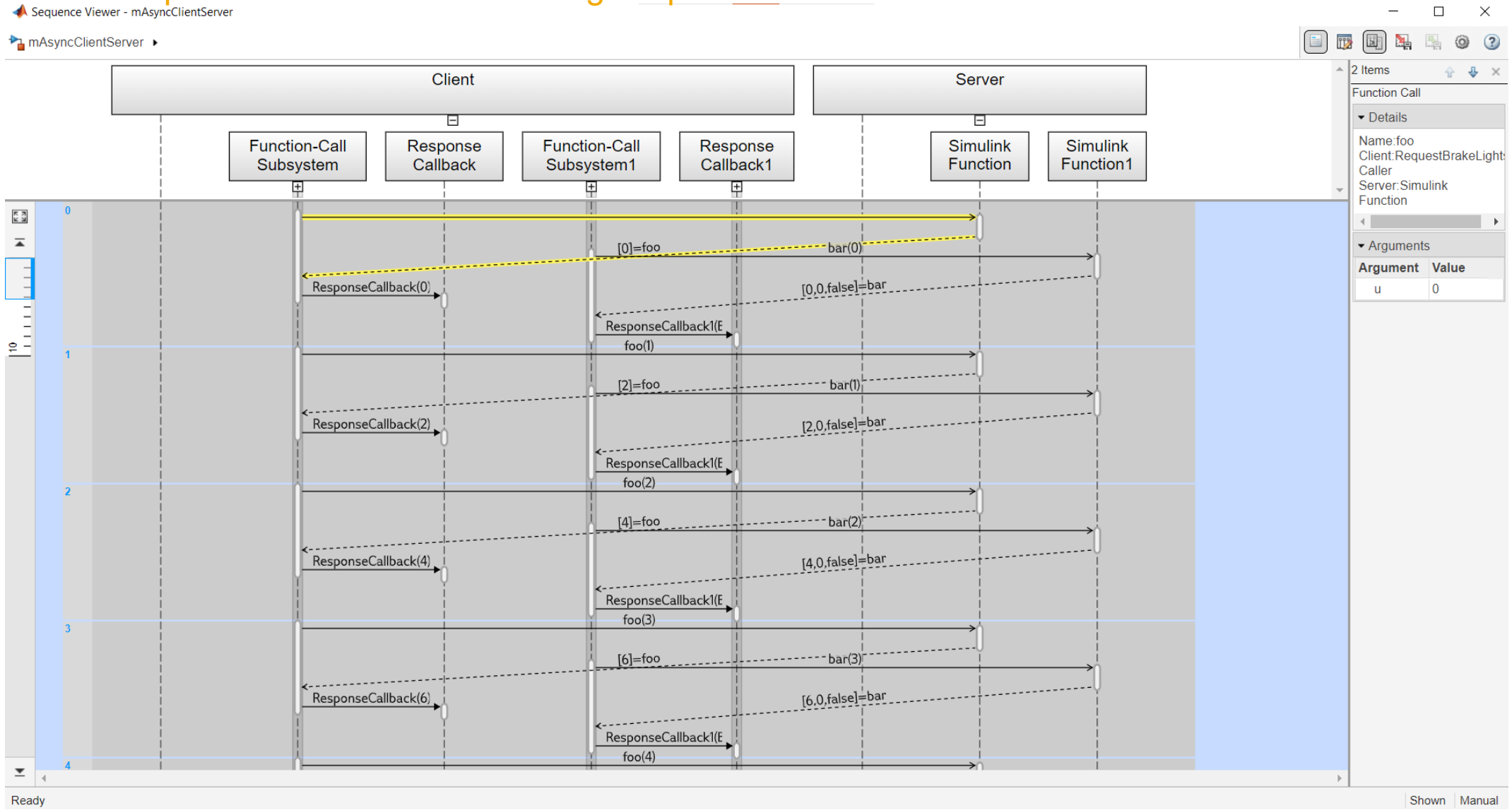
Service-Oriented Behavior



https://www.mathworks.com/help/simulink/ug/schedule-editor-overview.html

Service-Oriented Behavior Modeling

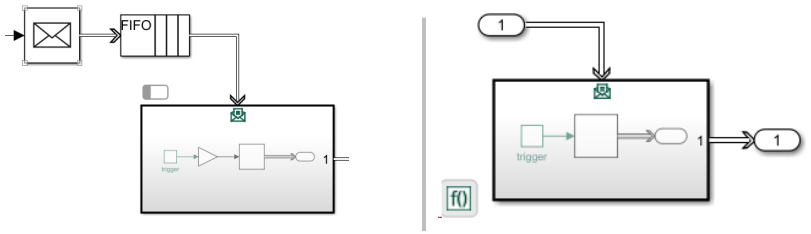
Visualize Sequence of Service Calls Using Sequence Viewer



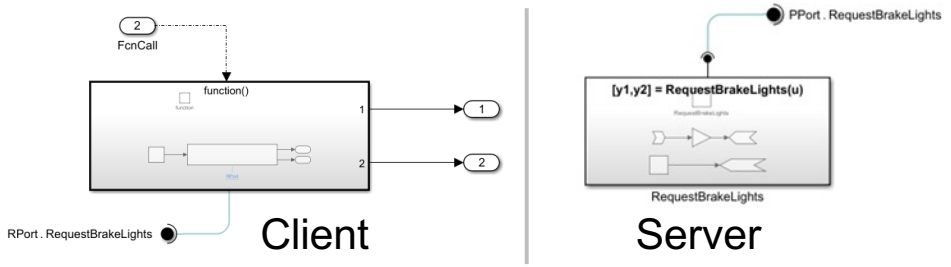
https://www.mathworks.com/help/systemcomposer/ref/sequenceviewertool.html

Behavior Modeling

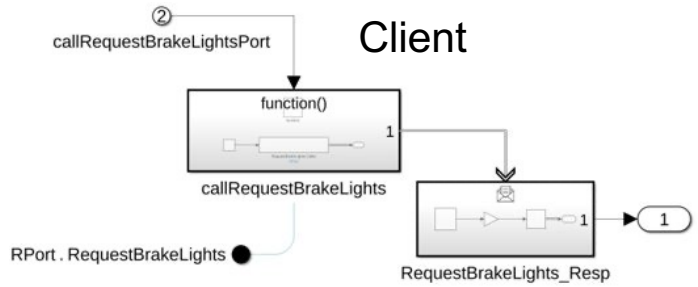
Message triggered subsystem



Synchronous client-server function port



Asynchronous client-server function port

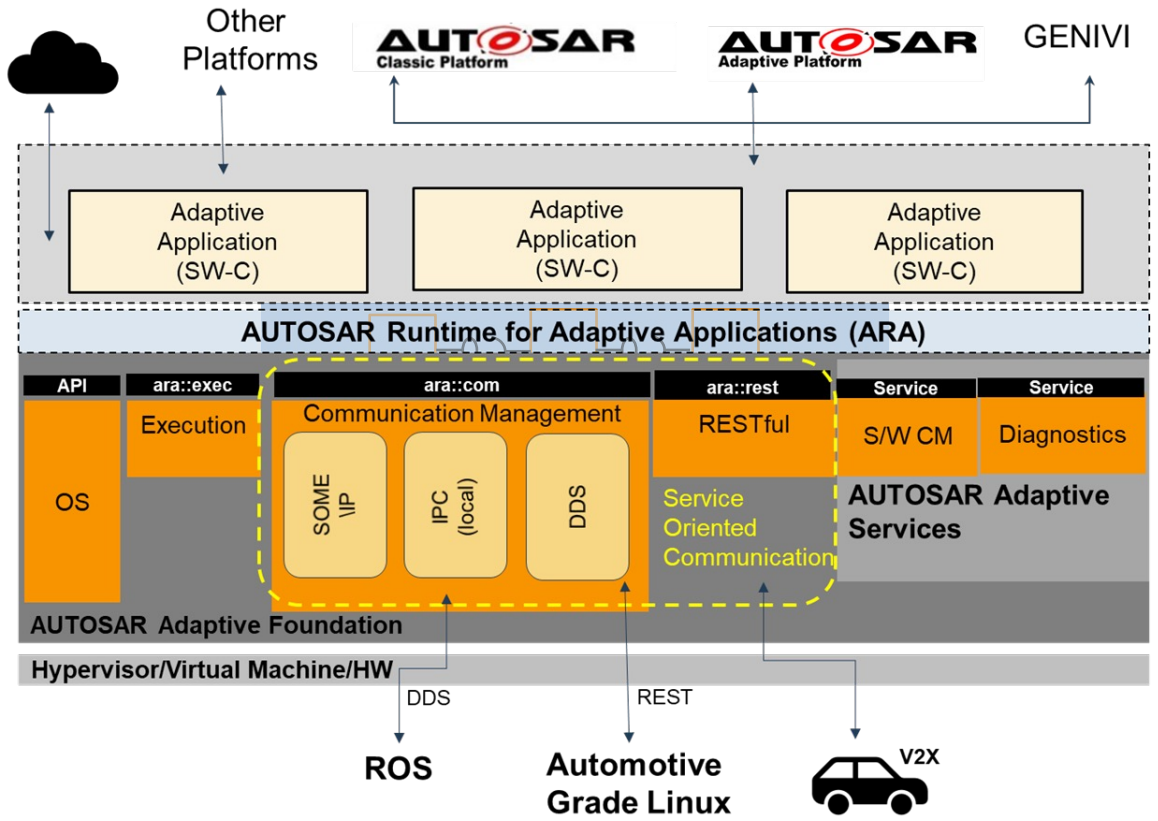


Agenda

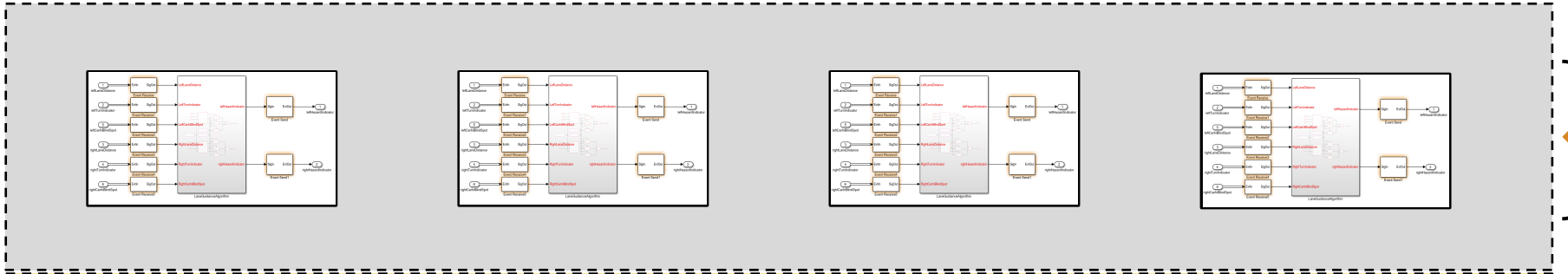
- Software-defined vehicles and new architectures (SOA)
- New Simulink Semantics
  - Generic SOA
  - SOA based AUTOSAR Adaptive applications
- Conclusions and key takeaways

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 Runtime for Adaptive Applications (ARA)

API
OS

API
Execution

API
Communication

Service
S/W CM

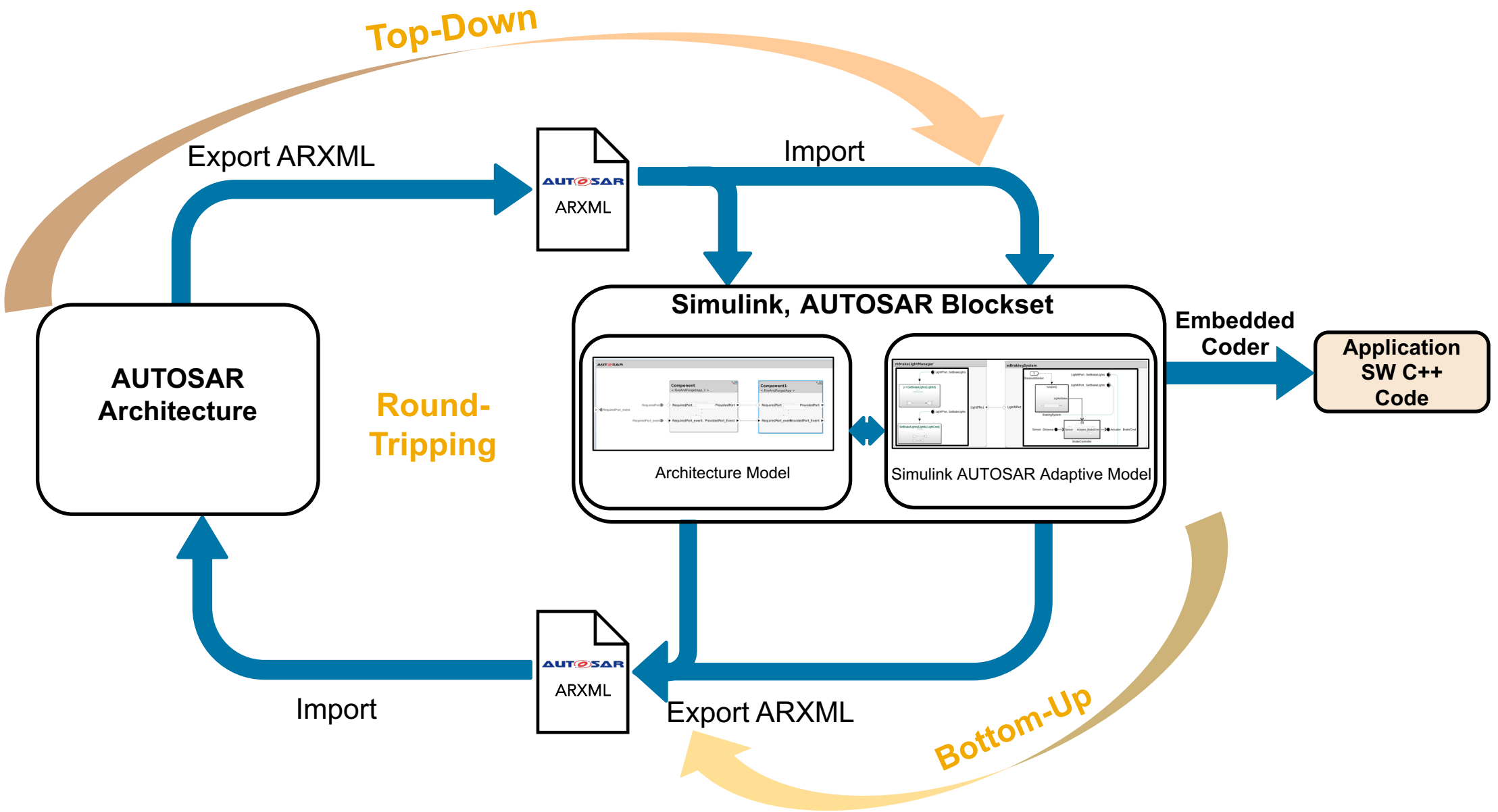
Service
Diagnostics

Adaptive AUTOSAR Services

Adaptive AUTOSAR Foundation

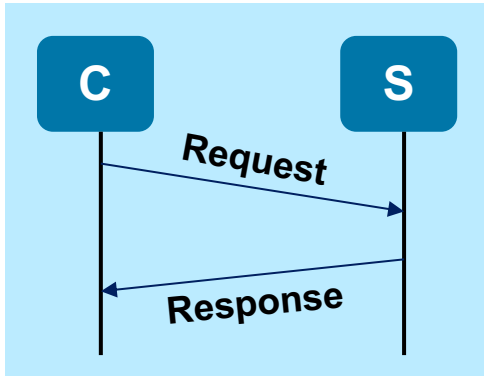
High Performance Hardware/Virtual Machine

AUTOSAR Workflows - Importing and Exporting AUTOSAR Descriptions for Adaptive applications

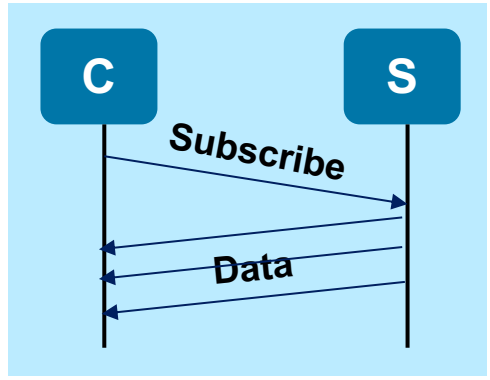


Service-oriented communication used in Adaptive applications

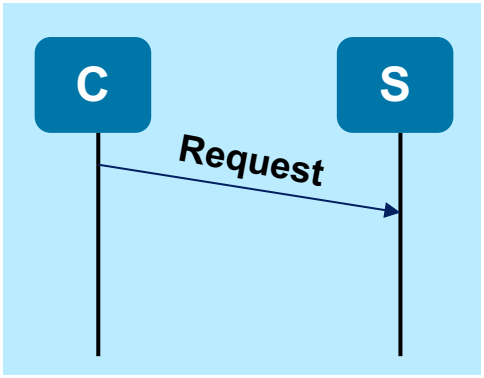
- Service Interface can contain
  - Methods
  - Events
  - Fields



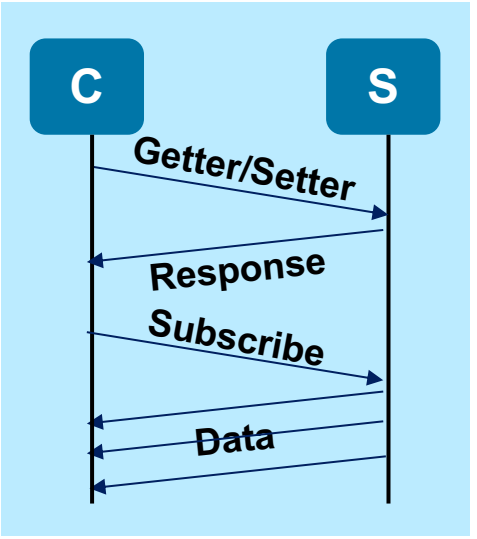
Method
Request/Response



Event

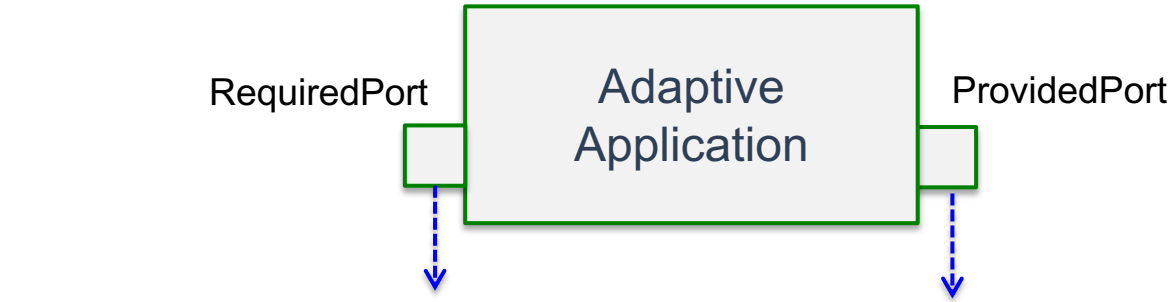


Method
Fire/Forget



Field

Adaptive SW Architecture Concepts



Service Interface

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

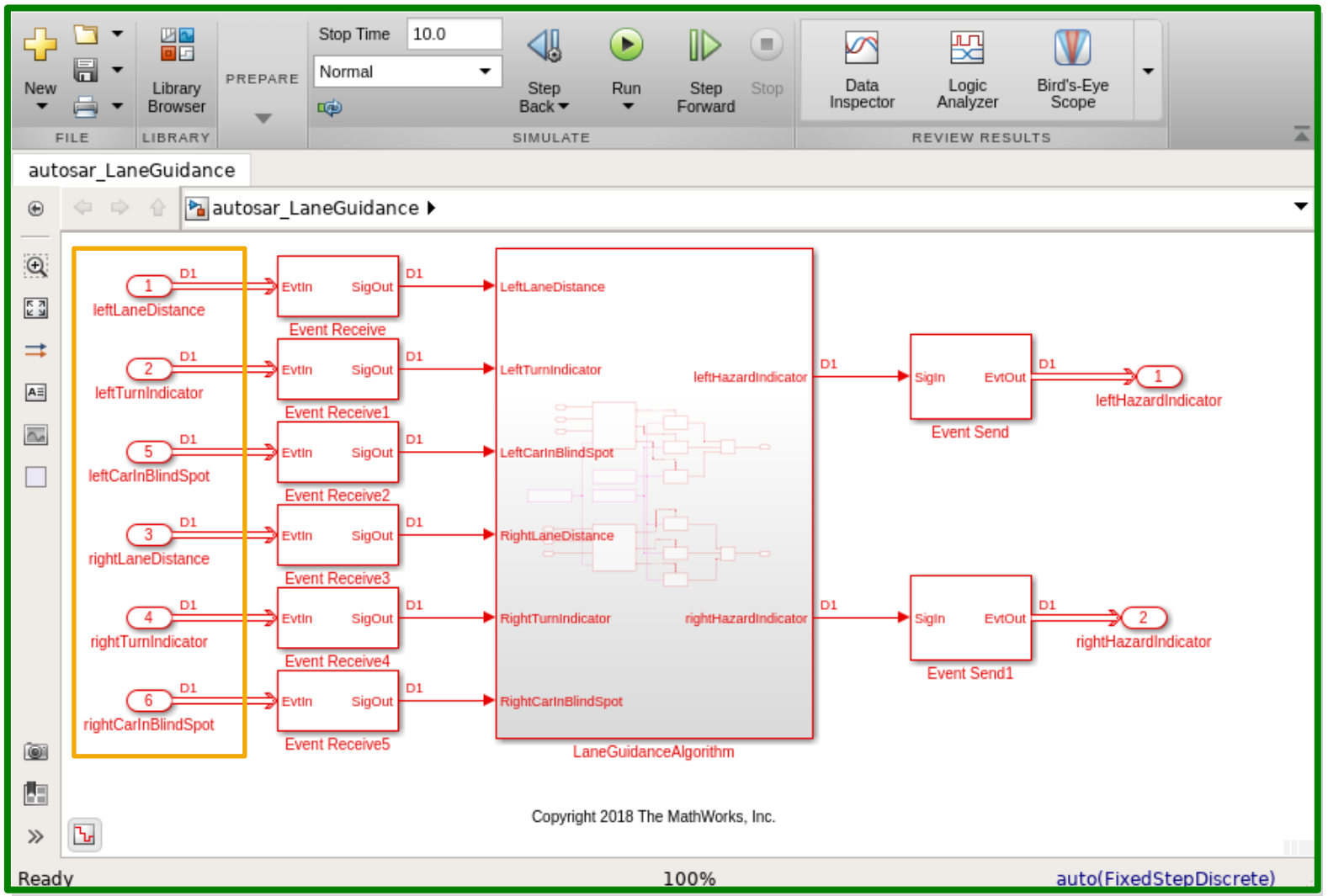
Mapping AUTOSAR Adaptive Concepts to Simulink

AUTOSAR

Adaptive Application

RequiredPort

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



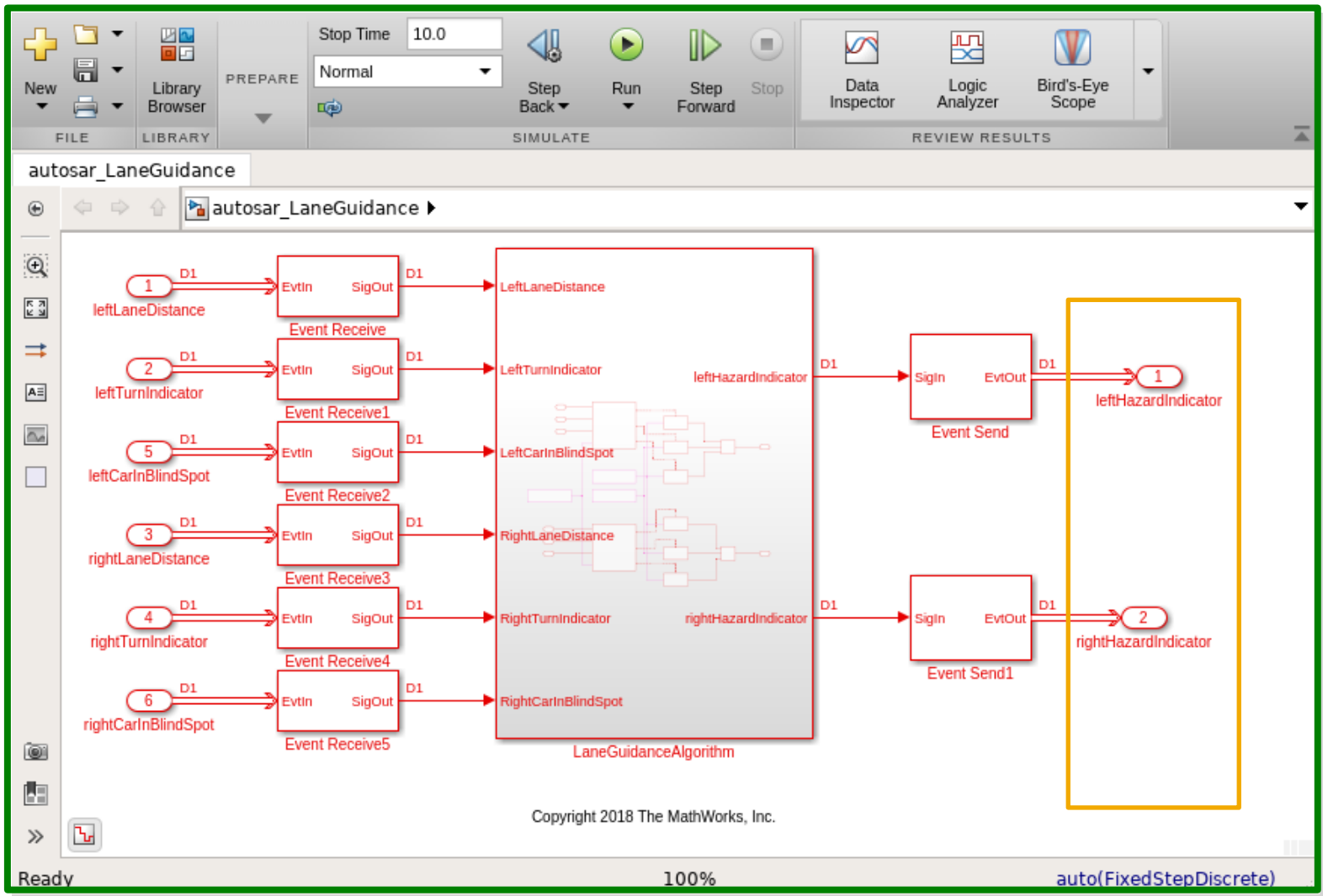
Mapping AUTOSAR Adaptive Concepts to Simulink

AUTOSAR

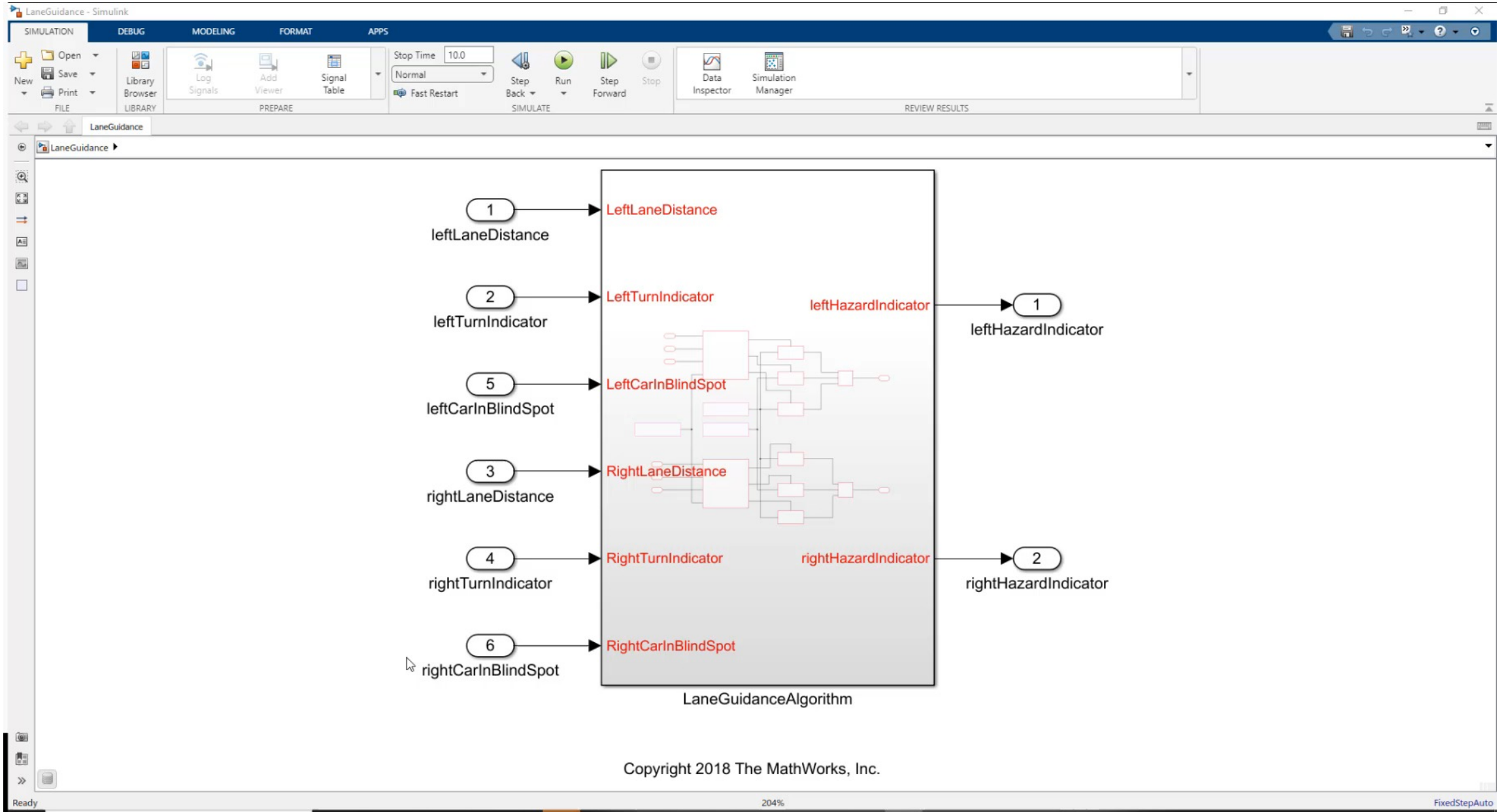
Adaptive Application

ProvidedPort

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

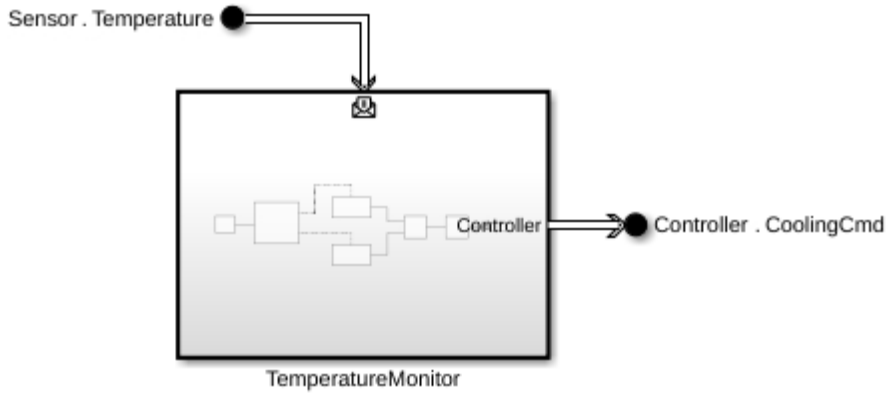


Configure for an AUTOSAR Adaptive application



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

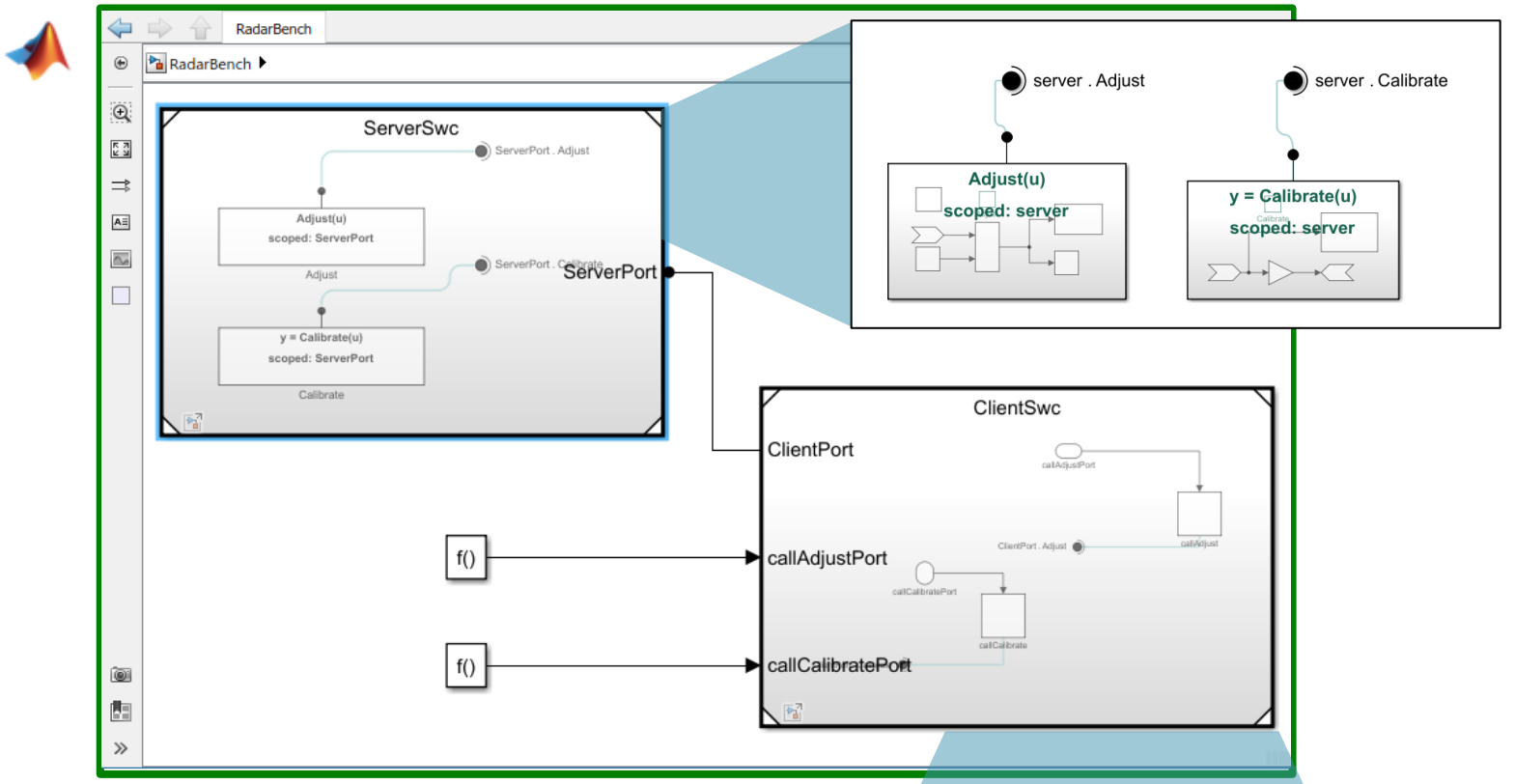


Events trigger the receiver execution

```
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);
}
```

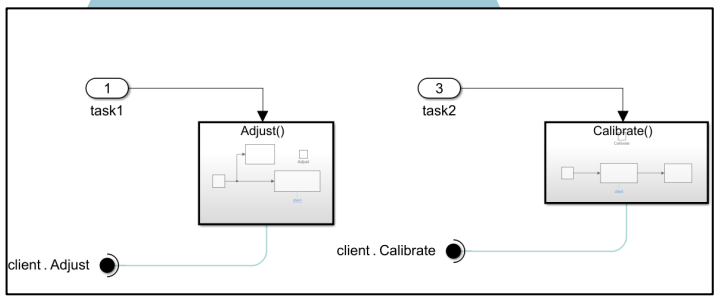
Modelling an AUTOSAR Adaptive application in Simulink


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



Modelling Methods

- Blocking Request-Response
- Fire-Forget methods

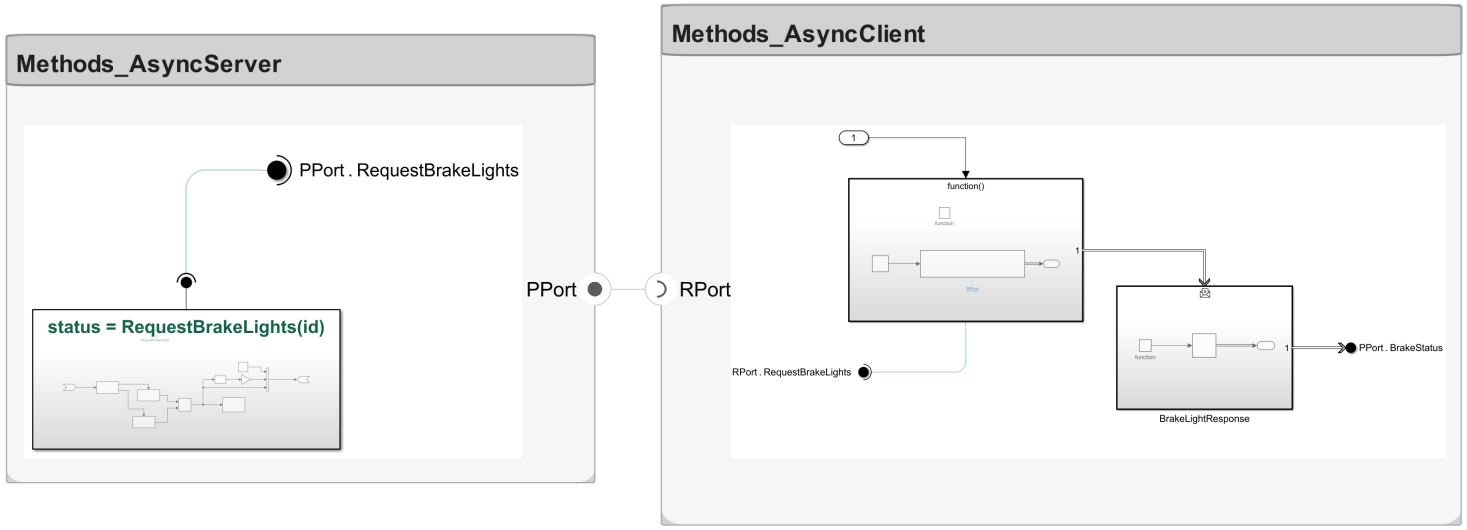


Model asynchronous/non-blocking methods

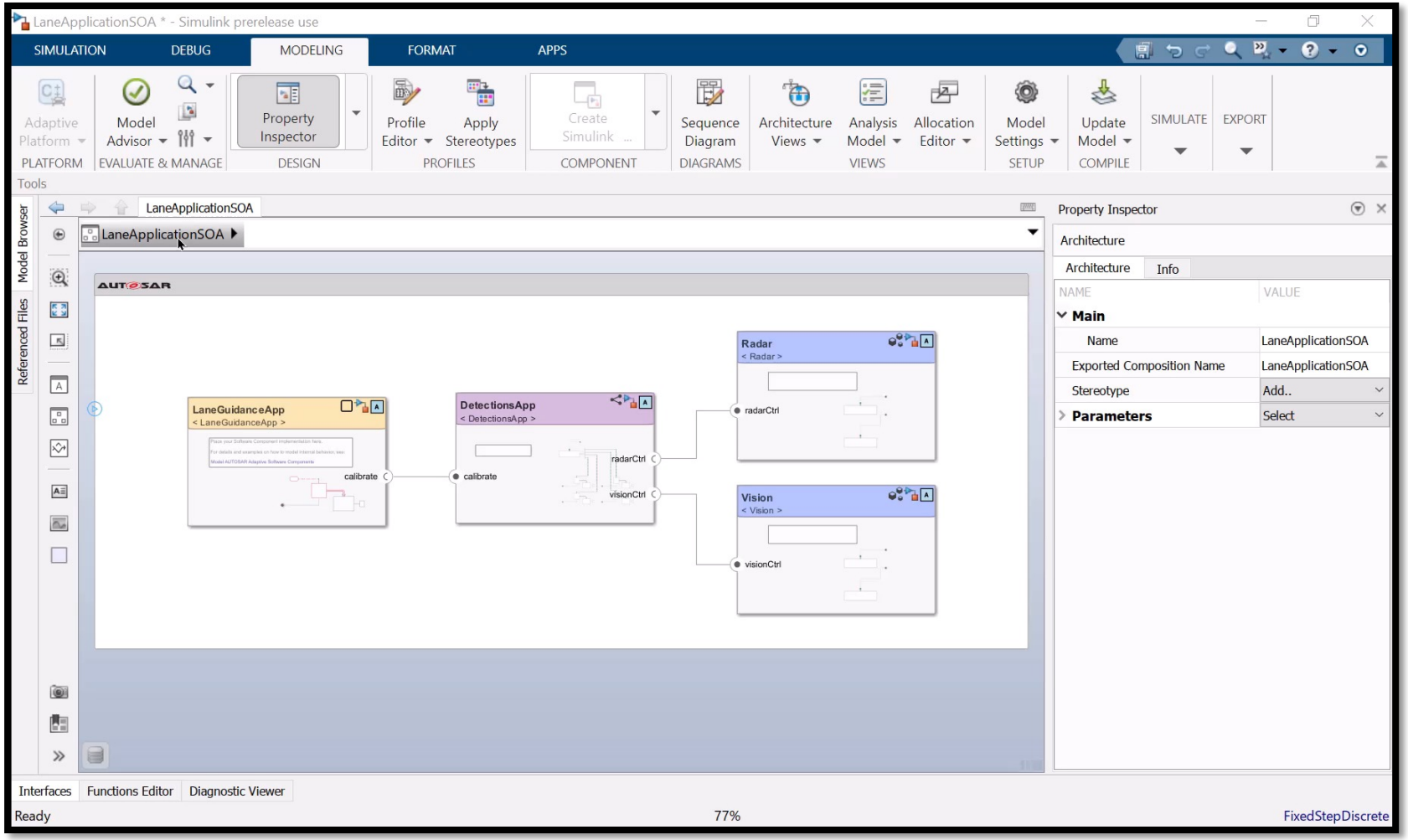
- Simulink supports modelling of methods
  - Non-blocking Request-Response methods

```
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



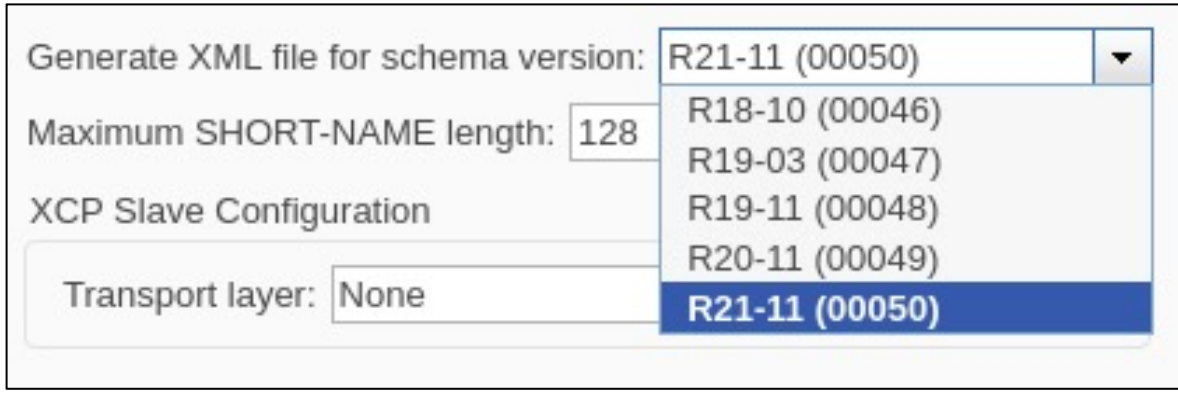
Authoring AUTOSAR Adaptive Application in action



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

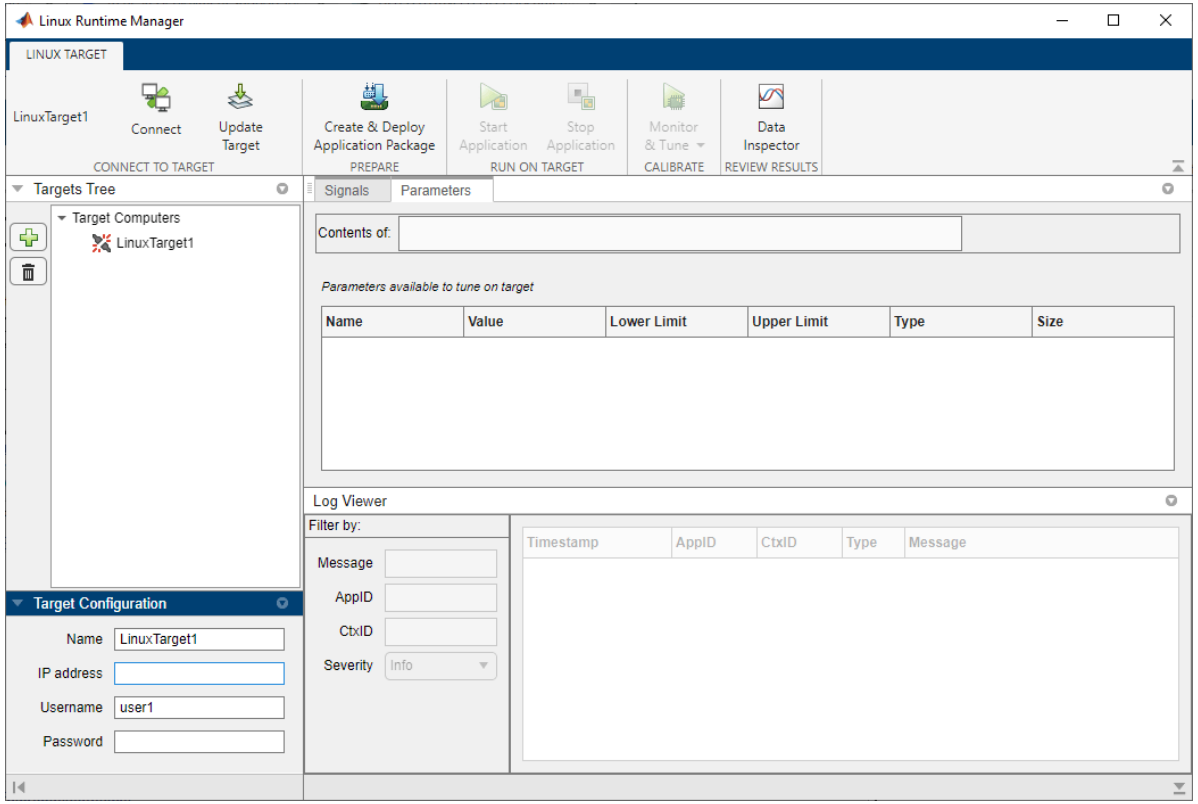
AUTOSAR Release	MathWorks Release
R18-10	R2019a
R19-03	R2020a
R19-11	R2021a
R20-11	R2022b
R21-11	R2023a

AUTOSAR Adaptive



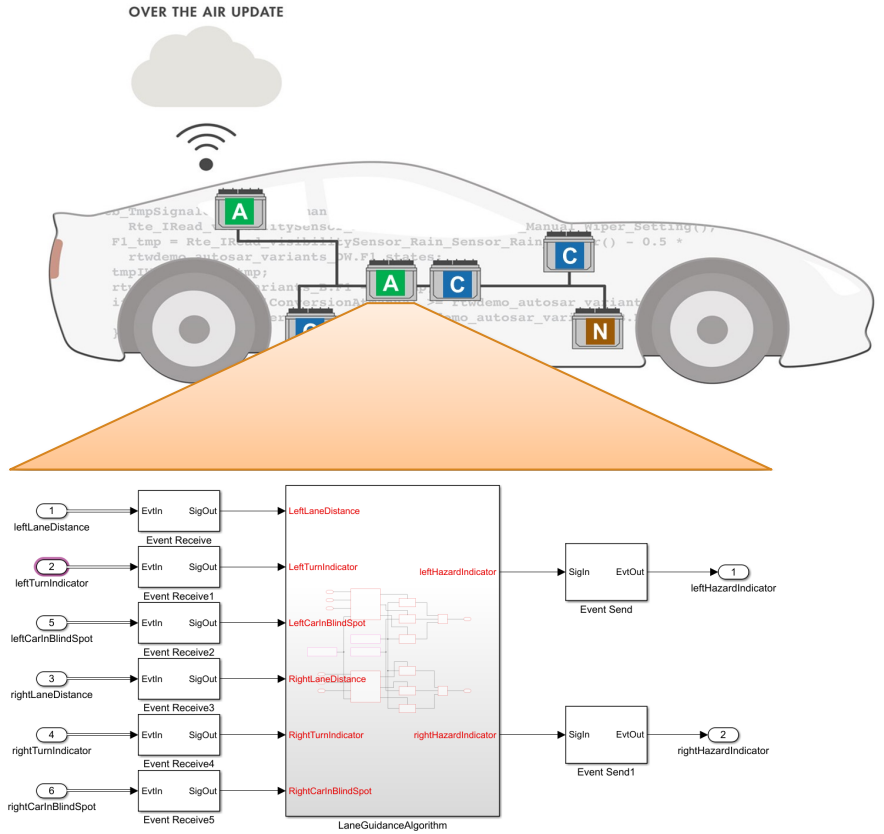
Deploy AUTOSAR Adaptive Architecture Models using Linux Runtime Manager

- 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

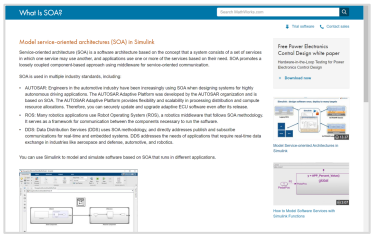
- Development of Adaptive applications require a **change of mindset**
- **Centralize, re-architect** existing applications and partition in processes and services

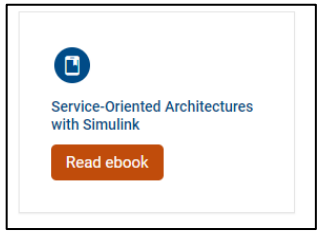
Solutions

- **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