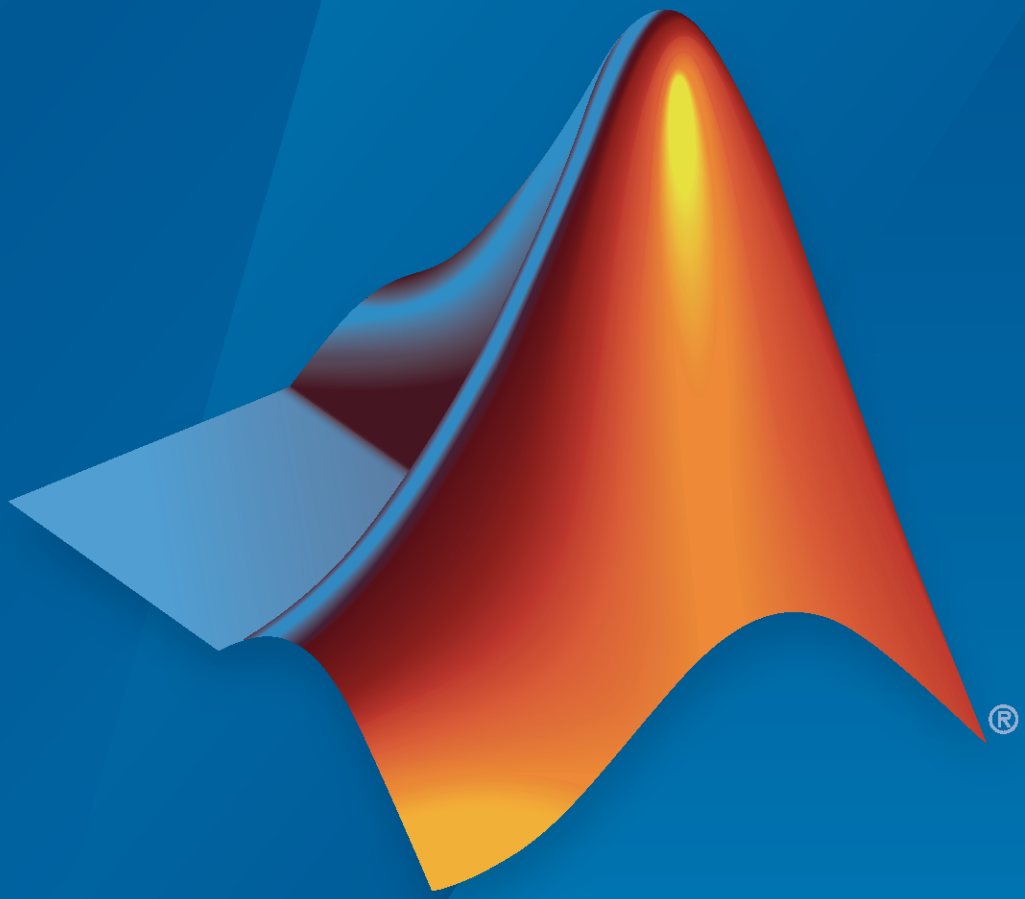
System Composer™ Release Notes



MATLAB® & SIMULINK®

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

System Composer™ Release Notes

© COPYRIGHT 2019–2022 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `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.

Patents

MathWorks products are protected by one or more U.S. patents. Please see `www.mathworks.com/patents` for more information.

Contents

R2022b

Author parameters on components that flow down through Simulink and architectures **1-2**

Authoring asynchronous client-server interfaces in software architecture models **1-3**

Enable functions authoring in AUTOSAR Blockset architectures **1-4**

Interface compatibility edit-time checks **1-4**

Edit, view, and manage allocations on architecture models **1-5**

Use stereotypes on requirements and links **1-5**

Simulate Sequence Diagrams: Use sequence diagrams to visualize System Composer architecture simulation **1-6**

Describe elements of a sequence diagram using annotations **1-6**

Export view diagrams **1-7**

Edit layout of architecture views interactively **1-7**

Allow merging of message lines in architecture models **1-7**

Report Generator: Generate reports for a System Composer architecture model **1-8**

R2022a

Instance-specific parameters for System Composer components **2-2**

Subsystem Reference Behaviors: Add reusable Simulink and Simscape behaviors to components **2-3**

Compare two versions of an architecture model using Comparison Tool **2-3**

Interface Adapter Bus Creation Mode: Use Adapter blocks to author interfaces for outgoing connections **2-4**

Synchronize sequence diagrams and architecture models **2-5**

Client-server interfaces: Model service-oriented architectures in software architecture models **2-5**

Author and edit functions in software architectures **2-6**

Combine multiple signal or message lines in software architectures into single line using the Adapter block **2-7**

Function Stereotypes: Apply stereotypes to functions of software architectures **2-9**

Software-in-the-loop (SIL) and processor-in-the-loop (PIL) simulation support for reference components **2-10**

Import and export functions of software architectures **2-11**

Author Model Advisor checks that run at edit-time **2-11**

R2021b

Physical Interfaces with Simscape: Create physical interfaces, ports, and connections on components **3-2**

Simulink Subsystem Component: Add Simulink and Simscape behaviors to components **3-2**

Create software architectures from existing components **3-3**

Functions Editor: Visualize component functions in software architectures **3-4**

Value Types as Interfaces: Describe atomic pieces of data **3-5**

Owned Interfaces: Define interfaces local to ports **3-6**

Architecture Views: Add port filters **3-6**

Architecture Hierarchy: Display the hierarchy of unique component types **3-7**

Class Diagrams: Display software architecture model as a class diagram **3-8**

Import and export software architectures **3-9**

Test harnesses for System Composer components **3-9**

R2021a

Sequence Diagrams: Describe system behavior as a sequence of interactions between components **4-2**

State Charts: Describe component behavior using Stateflow charts to represent modes of operation **4-2**

Software Architecture: Simulate and deploy software functions from System Composer **4-3**

Updates to the Architecture Views Gallery and Programmatic Interfaces **4-3**

Performance and Scalability Improvements **4-5**

Interface Editor enhancements **4-5**

R2020b

Model to Model Allocations: Establish traceable and directed relationships between architectural elements in source and target models **5-2**

Enhanced workflows with Architecture Views **5-2**

Hierarchy Views: Display component hierarchy as a tree diagram **5-3**

Referenced Data Dictionaries: Organize interfaces in a hierarchy of data dictionaries **5-3**

Interface Editor enhancements **5-3**

Support for component behavior using Simulink models with message input and output **5-4**

Stereotype-Based Styling: Style connectors based on stereotypes and use custom icons for component stereotypes **5-5**

Support for protected models as component behaviors **5-6**

Import and export requirement links along with the architecture models **5-6**

R2020a

Stereotype-Based Styling: Associate a color with component stereotypes **6-2**

Interface Stereotypes: Apply stereotypes for custom metadata on interface objects **6-2**

Quick Insert Stereotypes: Insert stereotyped components directly by typing on the canvas **6-3**

Model Templates for Referenced Models: Select from preconfigured templates when creating new referenced architecture or behavior models **6-4**

Requirements on Component Ports: Link requirements to component ports **6-4**

Partial Architecture Model Load: Improved loading for large models . . . **6-5**

R2019b

Architecture Views **7-2**

Interface Adapter **7-2**

Import and Export Architecture Models **7-2**

AUTOSAR Software Architecture **7-2**

R2019a

Introducing System Composer **8-2**

Composition Editor **8-2**

Spotlight Views **8-2**

Linking Simulink Behavior Models **8-2**

Linking and Managing Requirements **8-2**

Stereotypes and Profiles **8-2**

Architecture Model Analysis **8-3**

R2022b

Version: 2.3

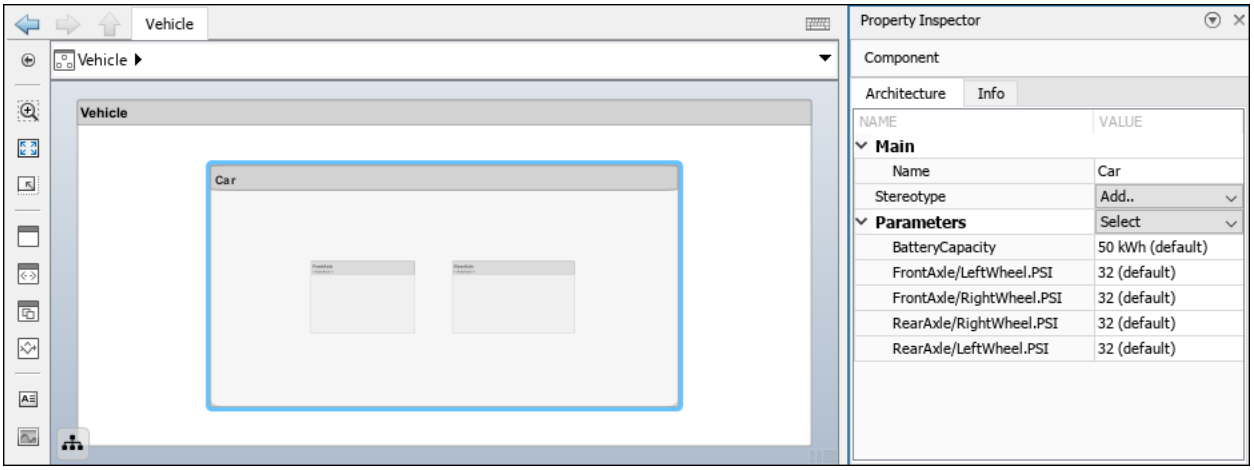
New Features

Bug Fixes

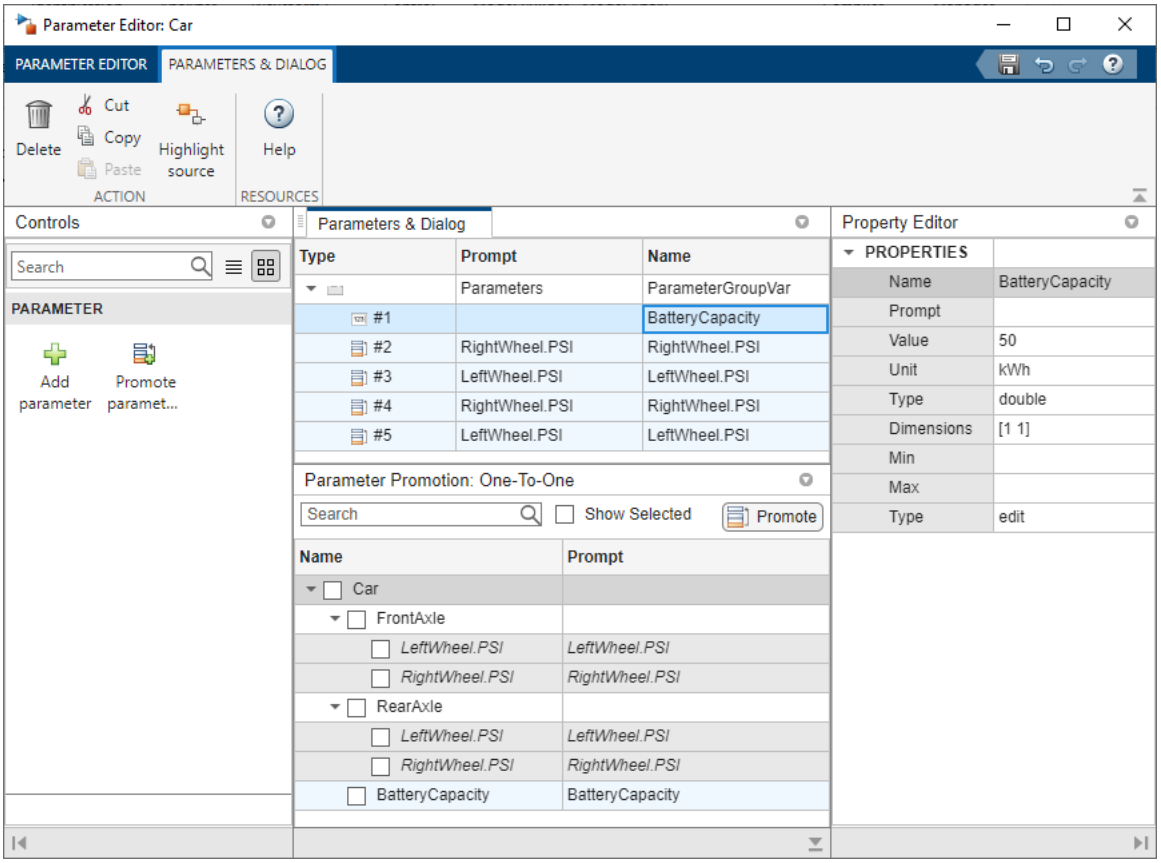
Compatibility Considerations

Author parameters on components that flow down through Simulink and architectures

Starting in R2022b, top-down parameter authoring is supported for System Composer architectures and AUTOSAR Blockset software architectures. Use the **Property Inspector** to view and edit parameters.



Use the **Parameter Editor** to author, edit, and manage parameters. Expose parameters to higher levels of the architectural hierarchy by promoting them.



Parameters are synchronized with Simulink® for simulation and code generation.

For more information, see “Author Parameters in System Composer Using Parameter Editor”.

Compatibility Considerations

This table shows the new and removed programmatic interfaces in R2022b.

New	Removed	Rationale
`systemcomposer.arch.Parameter`	`getParameterDefinition` and `systemcomposer.parameter.ParameterDefinition`	The new object `systemcomposer.arch.Parameter` has a `Type` property that is a `systemcomposer.ValueType` object with the parameter definition. The `Parameters` property under the `systemcomposer.arch.Architecture` object contains the `systemcomposer.arch.Parameter` object with the default parameter value in its `Value` property.

Authoring asynchronous client-server interfaces in software architecture models

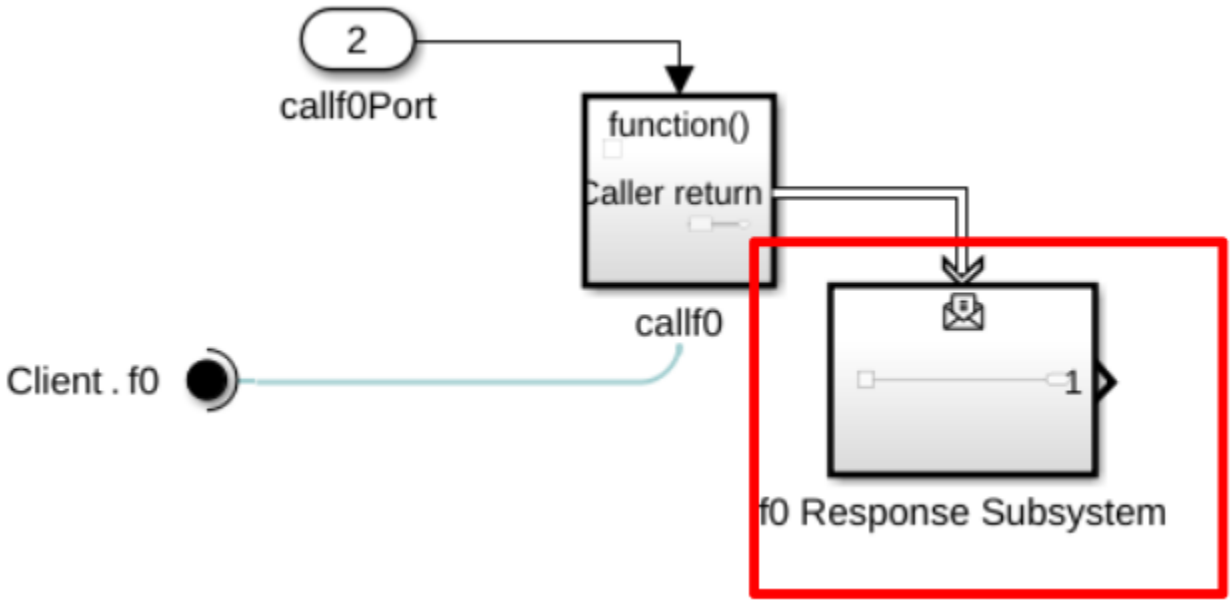
In R2022b, you can author asynchronous client-server interfaces in software architecture models.

To specify a function element as asynchronous:

- In the **Interface Editor**, in the new **Asynchronous** column, select the corresponding check box.

- Use the `setAsynchronous` function to make a `systemcomposer.interface.FunctionElement` object asynchronous.

Assign the service interface to client-server ports on software components. Create Simulink behaviors for these software components that include asynchronous Function Caller and Simulink Function blocks. A Message Triggered Subsystem block that connects to the output port of the Function Caller block acts as a callback for the function.



You can schedule the priority of asynchronous services using the **Functions Editor**. For more information, see “Author Service Interfaces for Client-Server Communication”.

Enable functions authoring in AUTOSAR Blockset architectures

In R2022b, functions authoring is enabled in AUTOSAR Blockset software architectures. The System Composer **Functions Editor** is now available from the toolstrip in the **Modeling** menu.

The **Functions Editor** updates the diagram and populates it with the functions (runnables) from the reference components of the architecture model.

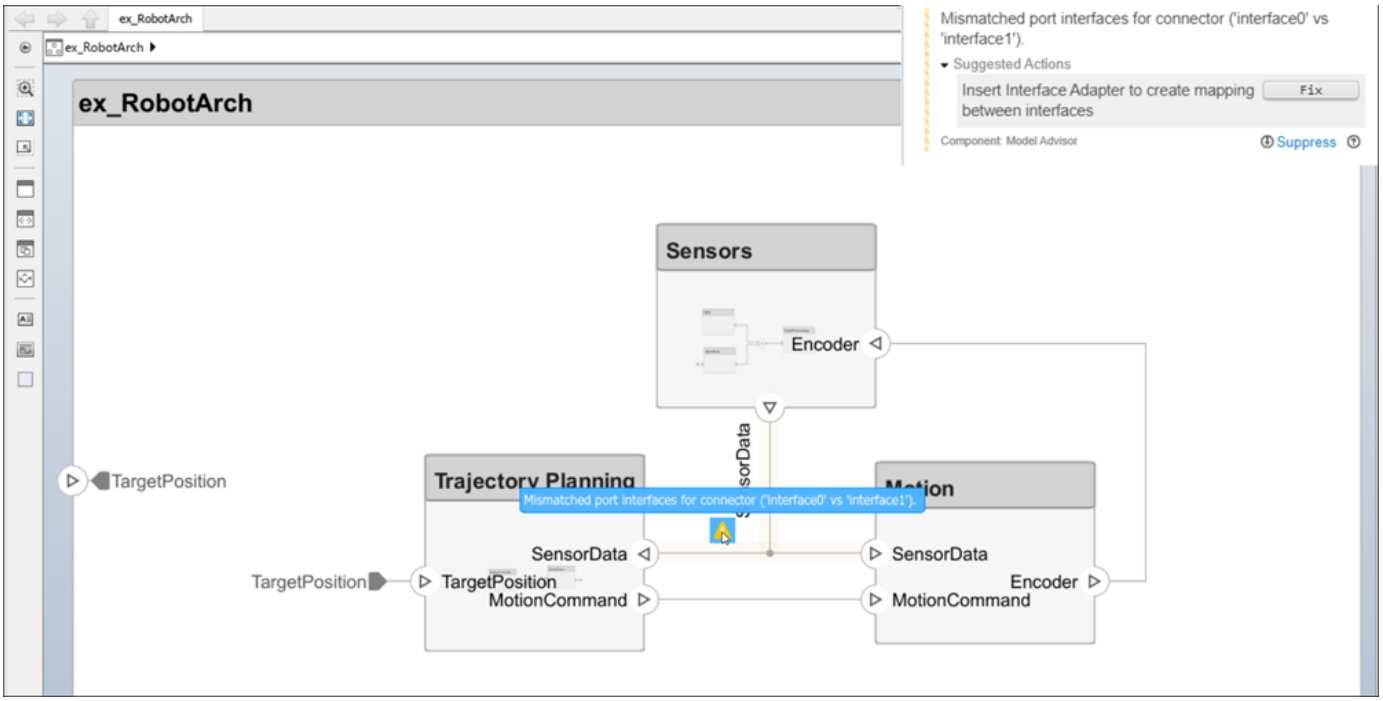
Compatibility Considerations

If an AUTOSAR architecture model created in a previous release is loaded in R2022b, and if it contains reference components, use the **Schedule Editor** to inspect and verify the partitions which correspond to functions. Otherwise, the partitions might not match ones from a previous release. Use the **Functions Editor** to specify the execution order of the functions.

Interface compatibility edit-time checks

In R2022b, if you have a Simulink Check™ license, you can enable the interface compatibility edit-time checks.

For incompatible interfaces on different ports on the same connection, such as different data interfaces, you can fix the problem by adding an Adapter block to define interface mappings.

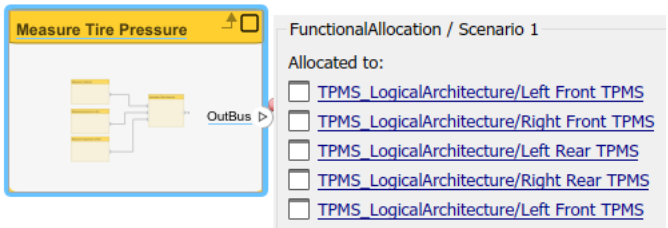


For more information, see “Enable Interface Compatibility Edit-Time Check”.

Edit, view, and manage allocations on architecture models

Starting in R2022b, you can manipulate allocations from within the model canvas.

- Create new allocations for components, ports, and connectors in architecture model editors by using the context menu.
- Select or create new allocation sets and scenarios on the Select Active Scenario dialog box.
- Inspect allocations from the composition editor using badges on allocated elements.



- Follow the allocation thread by navigating from one end to the other end of the allocation link.

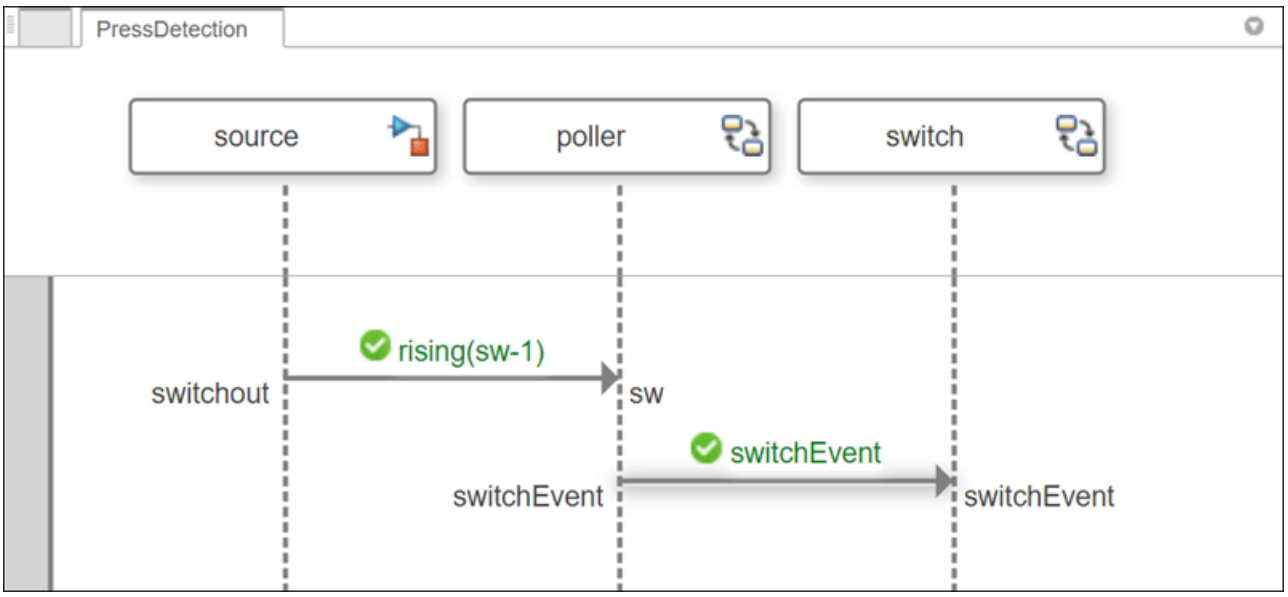
For more information, see “Create and Manage Allocations Interactively”.

Use stereotypes on requirements and links

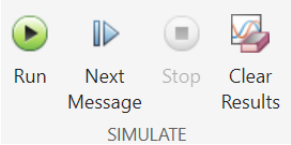
You can now apply stereotypes to requirements in a requirement set and to links in a link set in System Composer. A Requirements Toolbox™ license is required to author requirements. For more information, see “Customize Requirements and Links by Using Stereotypes” (Requirements Toolbox).

Simulate Sequence Diagrams: Use sequence diagrams to visualize System Composer architecture simulation

Starting in R2022b, from a sequence diagram in System Composer, you can simulate the parent architecture model and then verify that the model simulation results match the interactions described by the sequence diagram. Simulating a sequence diagram is not currently supported for software architectures.



To simulate sequence diagrams in the **Architecture Views Gallery**, on the **Sequence Diagram** tab, use the **Simulation** section of the toolstrip.



Alternatively, set the model configuration parameter `ObservedSequenceDiagrams`, then use the `sim` function.

For more information, see “Simulate Sequence Diagrams for Traffic Light Example”.

Describe elements of a sequence diagram using annotations

You can add plain-text annotations to a sequence diagram to describe diagram elements, such as lifelines, messages, and fragments.

For more information, see “Use Annotations to Describe Elements of Sequence Diagram”.

Export view diagrams

Starting in R2022b, you can print view diagrams in the **Architecture Views Gallery** using the **Export** button. Export the currently selected diagram as a PDF. View diagrams can be saved as PDF files. Sequence diagrams can be saved as PDF files or image files.

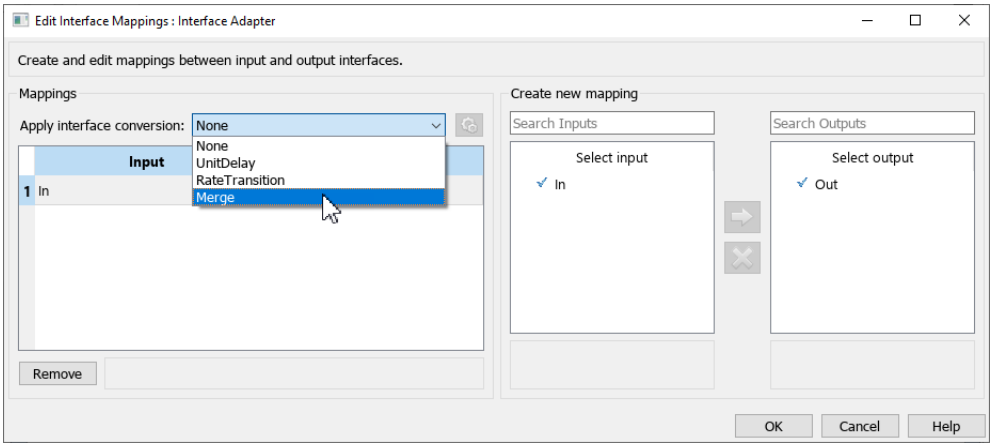
Edit layout of architecture views interactively

In the **Architecture Views Gallery**, you can now edit component diagram views interactively.

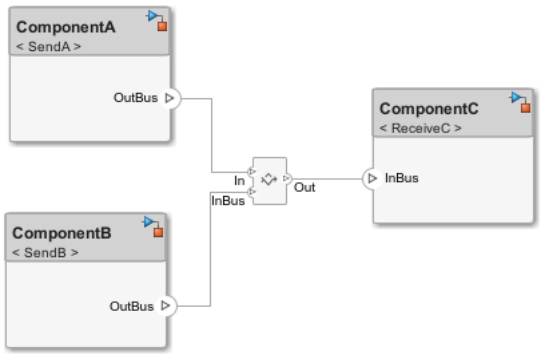
For more information, see “Edit Views Interactively”.

Allow merging of message lines in architecture models

You can now merge message lines in System Composer architecture models. Configure an Adapter block applying the interface conversion `Merge`.



Merge multiple messages produced by Simulink behavior components into a single destination using an Adapter block.



For more information, see “Merge Message Lines for Architectures Using Adapter Block”.

Report Generator: Generate reports for a System Composer architecture model

Using Simulink Report Generator™ classes, you can now generate custom reports for System Composer architectural data, and AUTOSAR Blockset software architectures. Customize your report using a set of report generator classes.

For more information, see “Import and Export Architecture Models”.

R2022a

Version: 2.2

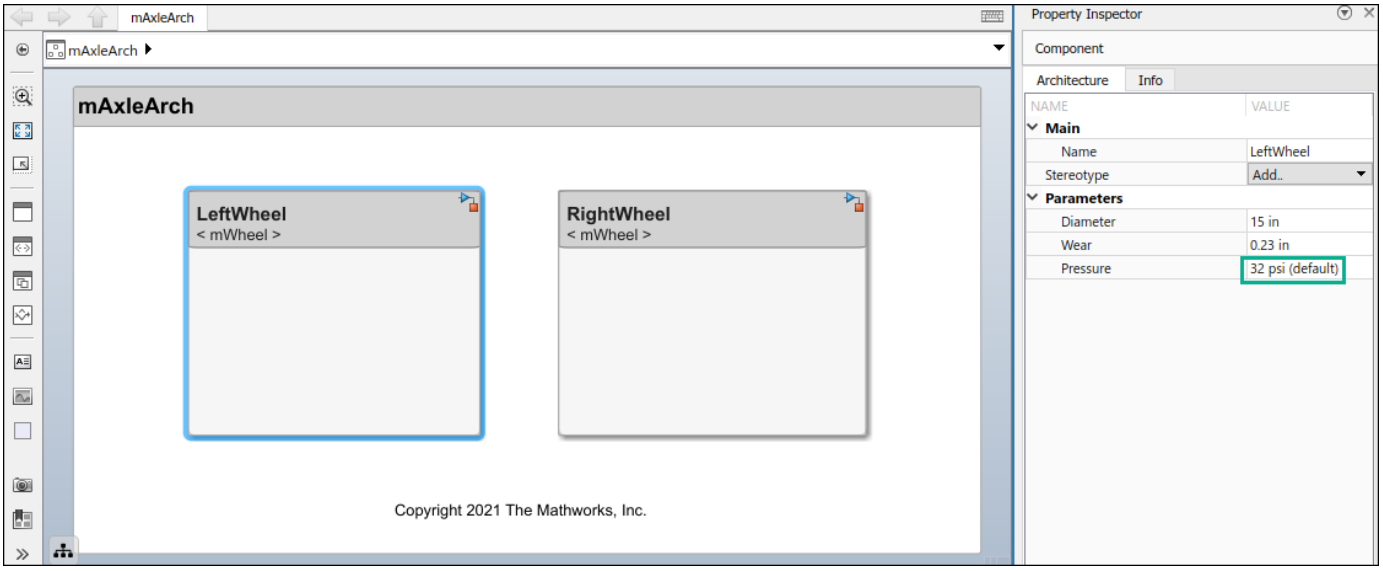
New Features

Bug Fixes

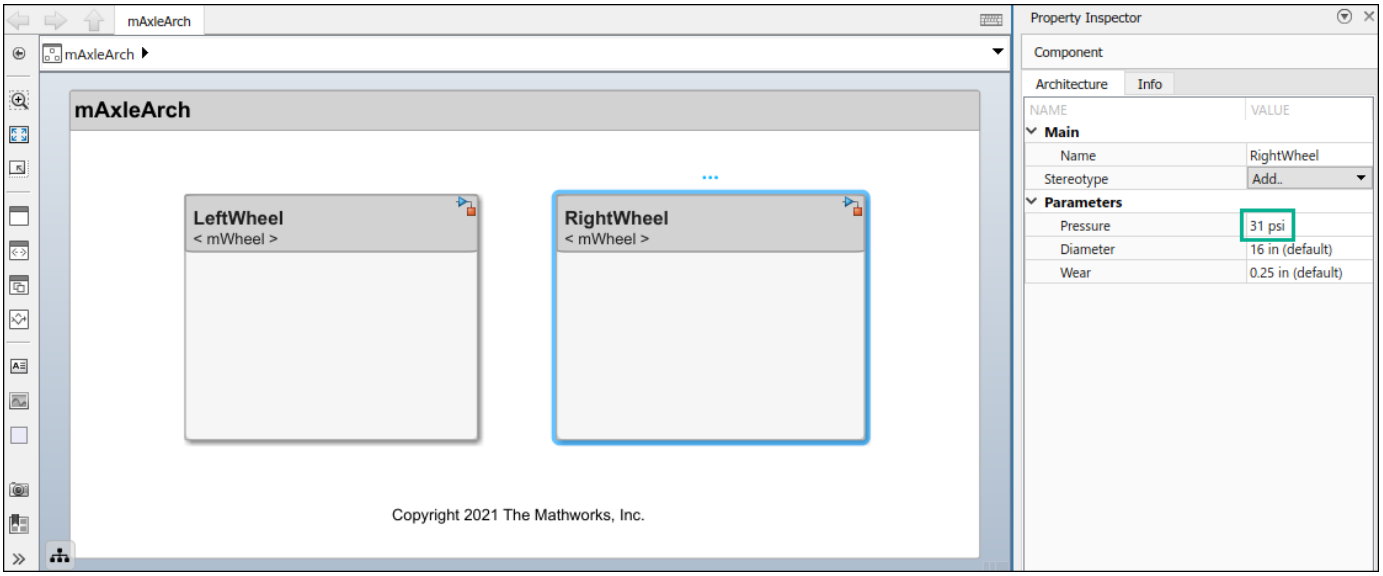
Instance-specific parameters for System Composer components

In R2022a, System Composer exposes instance-specific parameter values for reusable referenced models. You can now configure parameters defined by model arguments to use a different value for each instance of a component that references the same Simulink model or System Composer architecture model.

Instance-specific parameter values are now visible on the component level. View and edit these values using the Property Inspector.



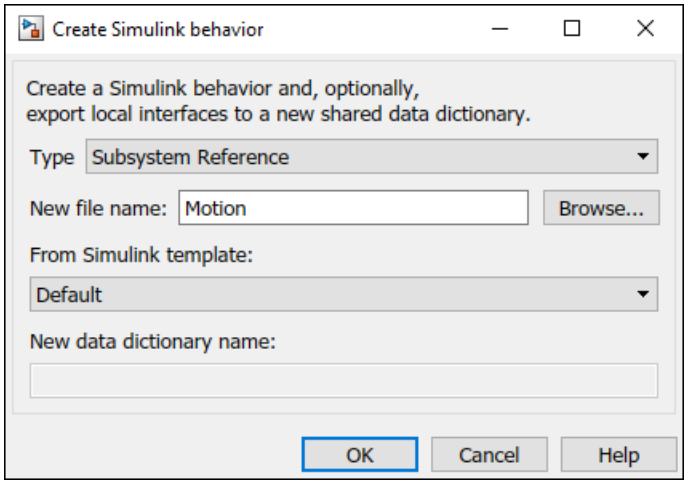
Each parameter value can be specified independently for each component that references the model.



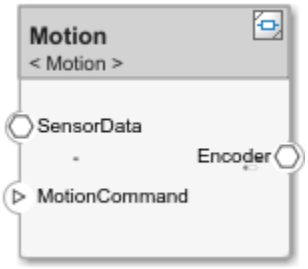
For more information, see Access Model Arguments as Parameters on Reference Components.

Subsystem Reference Behaviors: Add reusable Simulink and Simscape behaviors to components

Create a subsystem reference by linking a Simulink subsystem file to a component. The subsystem reference is saved in a separate file from the parent System Composer architecture model.



A subsystem reference component can be implemented as a reference to a separate Simulink subsystem file and reused multiple times in the architecture model. A subsystem reference component is different from a Simulink subsystem component that is part of the parent System Composer architecture model.



A common use case for subsystem reference is to author Simscape™ behaviors with physical ports, interfaces, connections, and blocks.

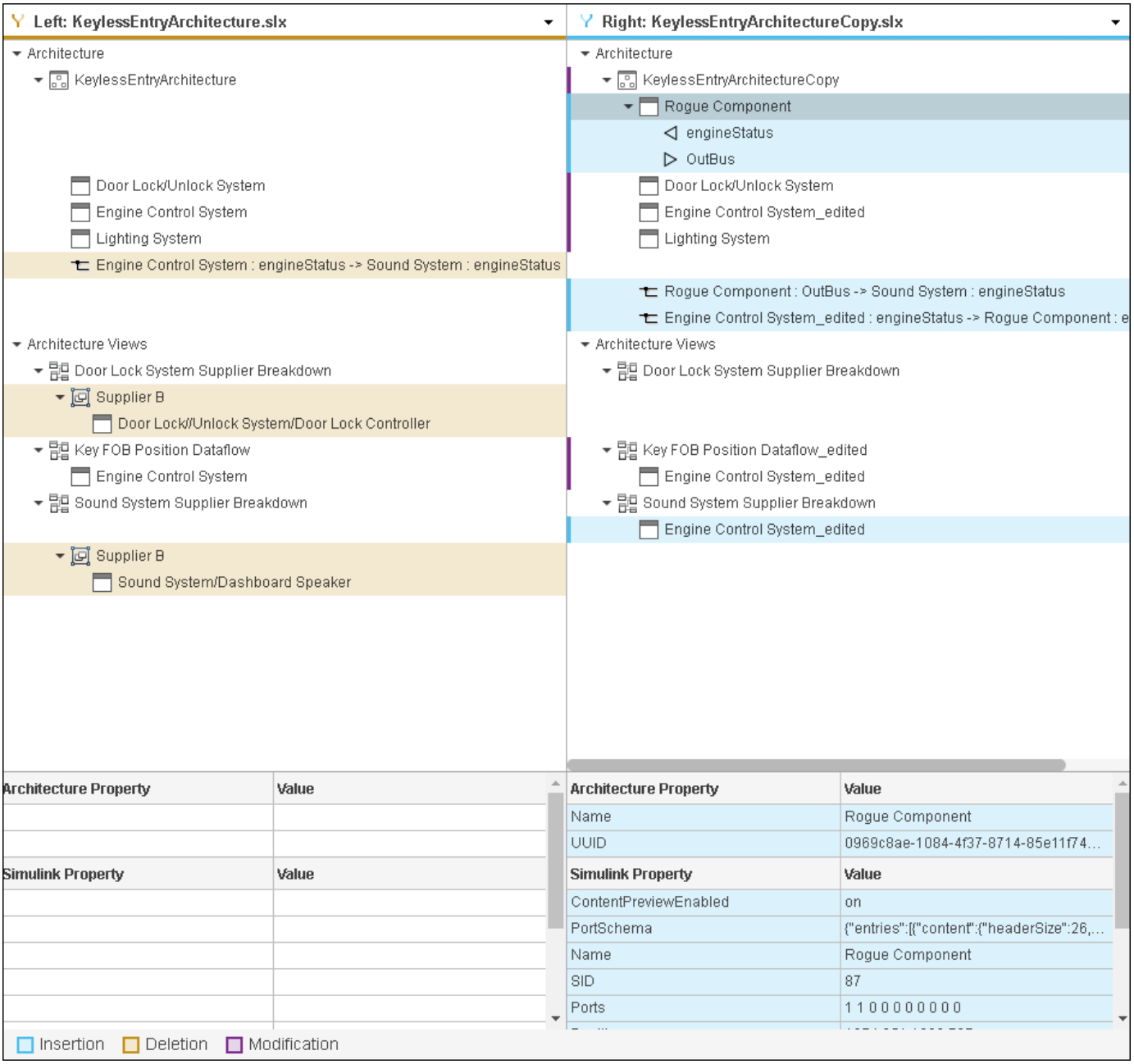
For more information, see Create Reusable Simulink Subsystem Behavior Using Subsystem Reference Component.

To convert a subsystem component into a subsystem reference, see Convert Simulink Subsystem Component to Subsystem Reference Component.

Compare two versions of an architecture model using Comparison Tool

In R2022a, you can compare two architecture models and see differences between architectural data using the **Comparison Tool**.

This example comparison shows the differences between two architecture models.



The System Composer Comparison Tool can also be used with software architectures in System Composer and AUTOSAR Blockset.

For more information, see Compare Model Differences Using System Composer Comparison Tool.

Interface Adapter Bus Creation Mode: Use Adapter blocks to author interfaces for outgoing connections

When input ports for an Adapter block are typed by interfaces from incoming connections and no interfaces are defined on the output ports, you can now use these interface elements to author owned

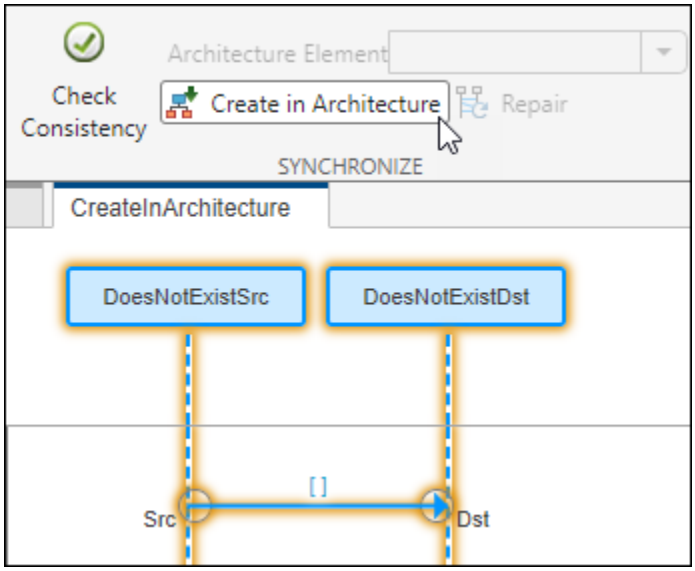
interfaces for outgoing connections. Instead of pre-defining interface structures, you can create the bus structure as you work. Double-click the Adapter block to open the Interface Adapter dialog in bus creation mode.

For more information, see Interface Adapter.

Synchronize sequence diagrams and architecture models

When an architecture model and sequence diagram are out of sync, you can now reconcile the differences and resolve warnings. Sequence diagrams can be edited from the **Architecture Views Gallery**. Click **Check Consistency** to check whether the sequence diagram is in sync. You can synchronize sequence diagrams and architecture models using these three actions:

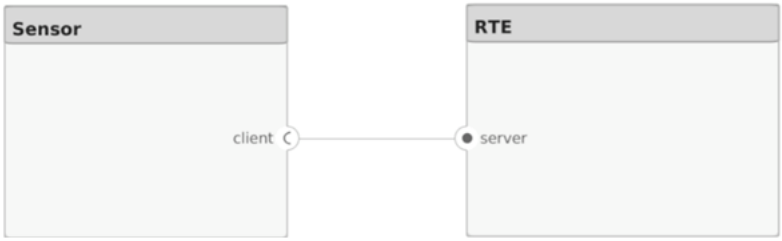
- Choose another **Architecture Element** from the list so that the selected lifeline points to the correct component or selected message end points to the correct port.
- Push changes from the sequence diagram to the architecture model using **Create in Architecture**.
- Pull changes from the architecture model to the sequence diagram using **Repair**.



For more information, see Synchronize Sequence Diagrams and Architecture Models.

Client-server interfaces: Model service-oriented architectures in software architecture models

You can now model client-server connections between software components in software architectures in System Composer. Client ports and server ports are new types of ports that you can add to software components. Service interfaces are a new type of interface you can associate with these ports.



A service interface defines the functional interface between client and server components. Each service interface consists of one or more function elements. Use the **Interface Editor** to author and edit service interfaces. Once you have defined a service interface, you can assign it to client and server ports using the Property Inspector. You can also use the Property Inspector to assign stereotypes to service interfaces.

To implement the desired function behavior for client and server components using referenced Simulink models, invoke the `Create Simulink behavior` command or the `createSimulinkBehavior` function for a component with a client or server port. You can also link existing Simulink models to components in System Composer to implement server and client behavior as long as the models are export-function models.

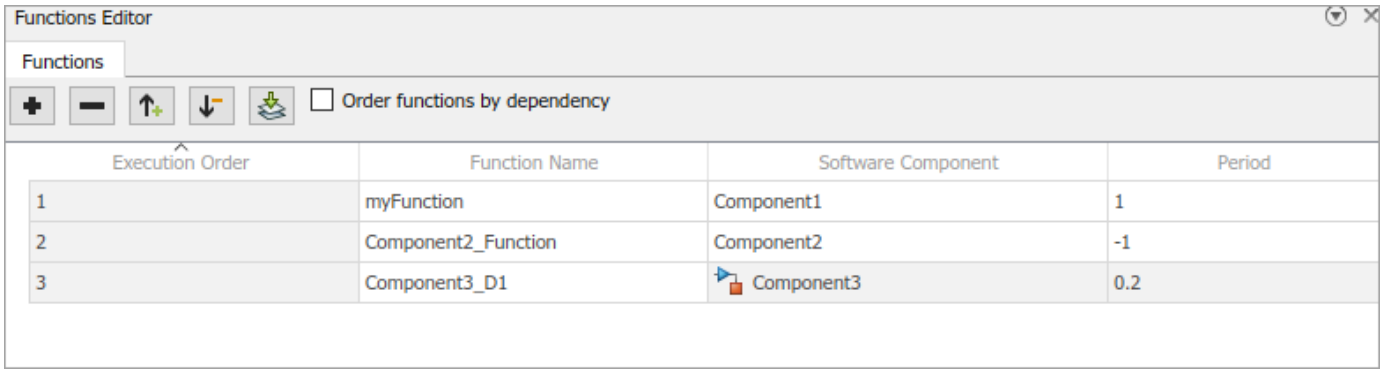
Once the behavior models are complete, you can simulate the composition model in System Composer. You can customize the execution order of client calls and server responses in a software architecture model using the Schedule Editor or the **Functions Editor**. After simulation, you can use the Sequence Viewer tool to visualize calls and responses.

For more information, see Author Service Interfaces for Client-Server Communication.

Author and edit functions in software architectures

In R2022a, you can author and edit functions for your software architecture components using the **Functions Editor** or programmatic interfaces. You can then implement authored functions by creating Simulink behaviors.

- Use the **Functions Editor** to author functions.
  - Add and delete functions.
  - Change the execution order of the functions.
  - Change the name of a function.
  - Change the period of a function.

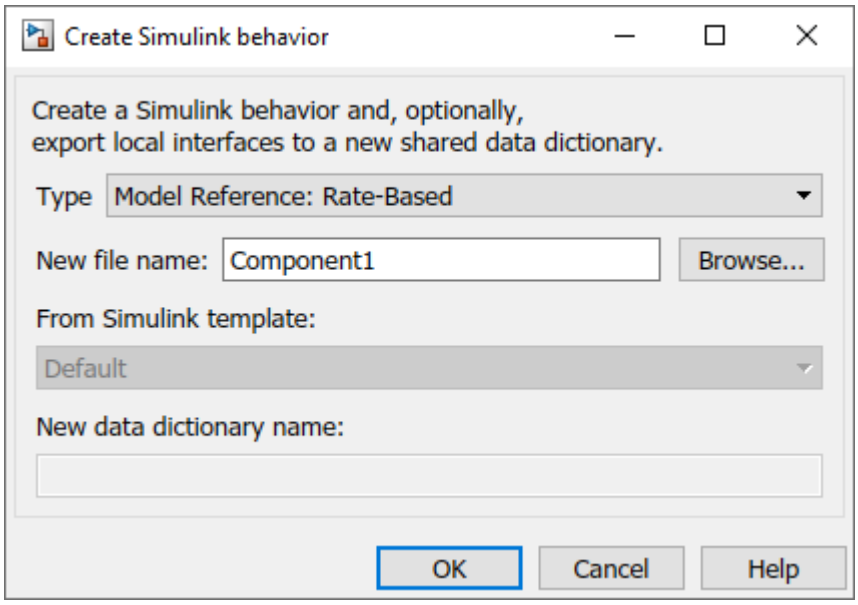


- You can also author functions for your components using the programmatic interface `addFunction`.

After you create your functions, you can implement and create behaviors for your functions using the toolstrip or a programmatic interface.

- To implement functions using the toolstrip:

  1. On the **Modeling** tab, select **Component**, then select **Create Simulink Behavior**.
  2. Select the **Type** of the Simulink behavior as rate-based or export-function.



     Alternatively, you can right-click the component and select **Create Simulink Behavior**.

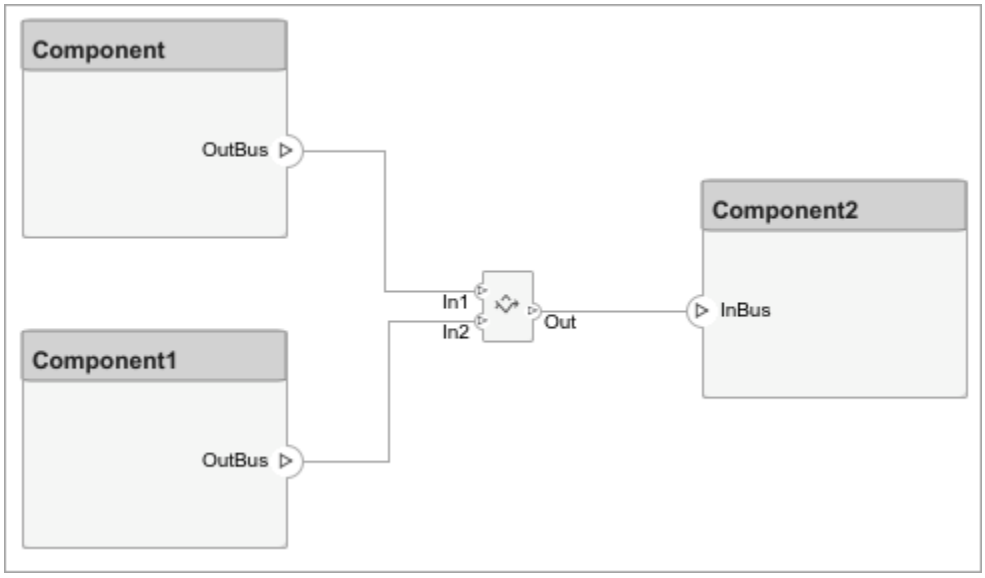
- You can also use the `createSimulinkBehavior` function to implement functions programmatically. This function creates a new Simulink rate-based or export-function behavior and links the software component to the new model.

For more information, see Author and Extend Functions for Software Architectures.

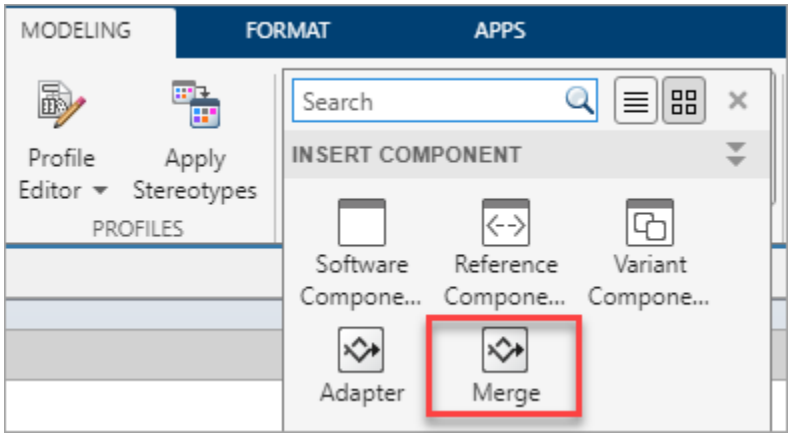
Combine multiple signal or message lines in software architectures into single line using the Adapter block

In R2022a, the Adapter block is enhanced to merge multiple signal or message lines into a single line.

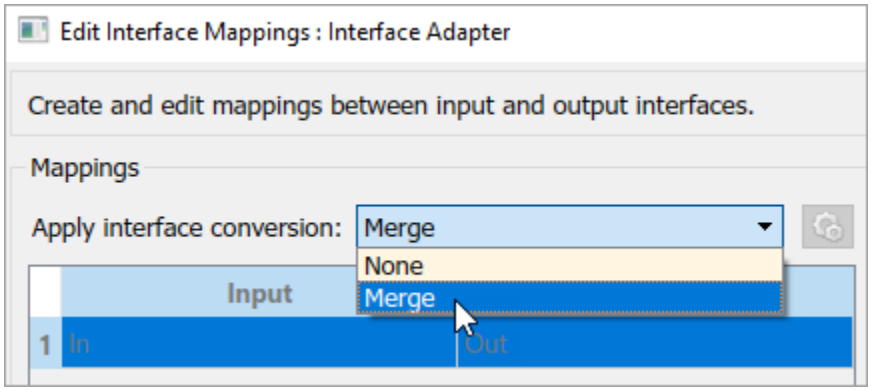
In this example, the Adapter block is configured to merge two signals from two different components.



To merge signal or message lines, you can add a Merge block from the toolstrip. The Merge block is an Adapter block preconfigured for merging.



You can also merge message or signal lines by setting the **Apply Interface conversion** parameter of the Adapter block to `Merge`.

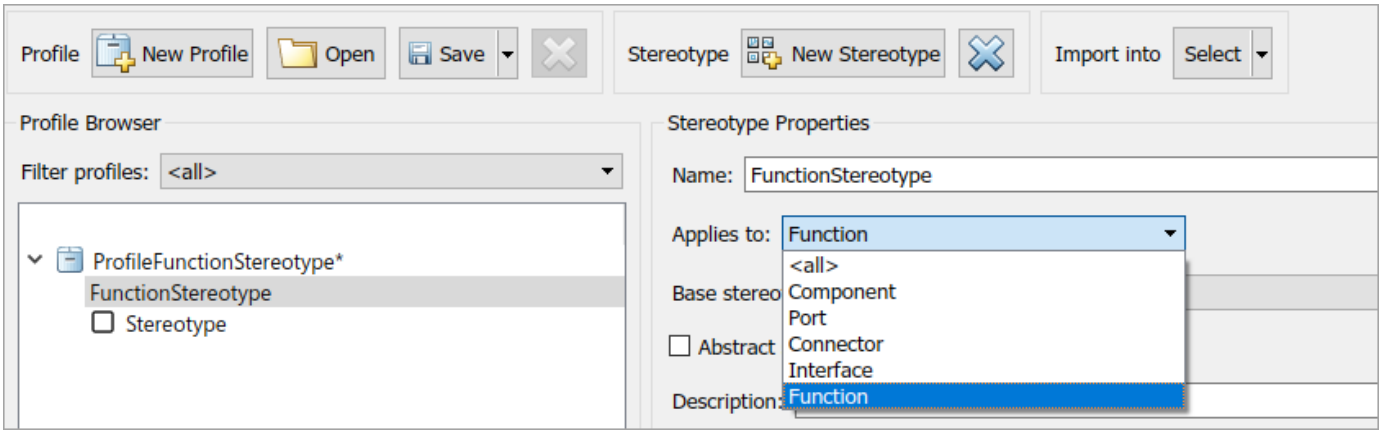


For more information, see Merge Message Lines Using Adapter Block.

Function Stereotypes: Apply stereotypes to functions of software architectures

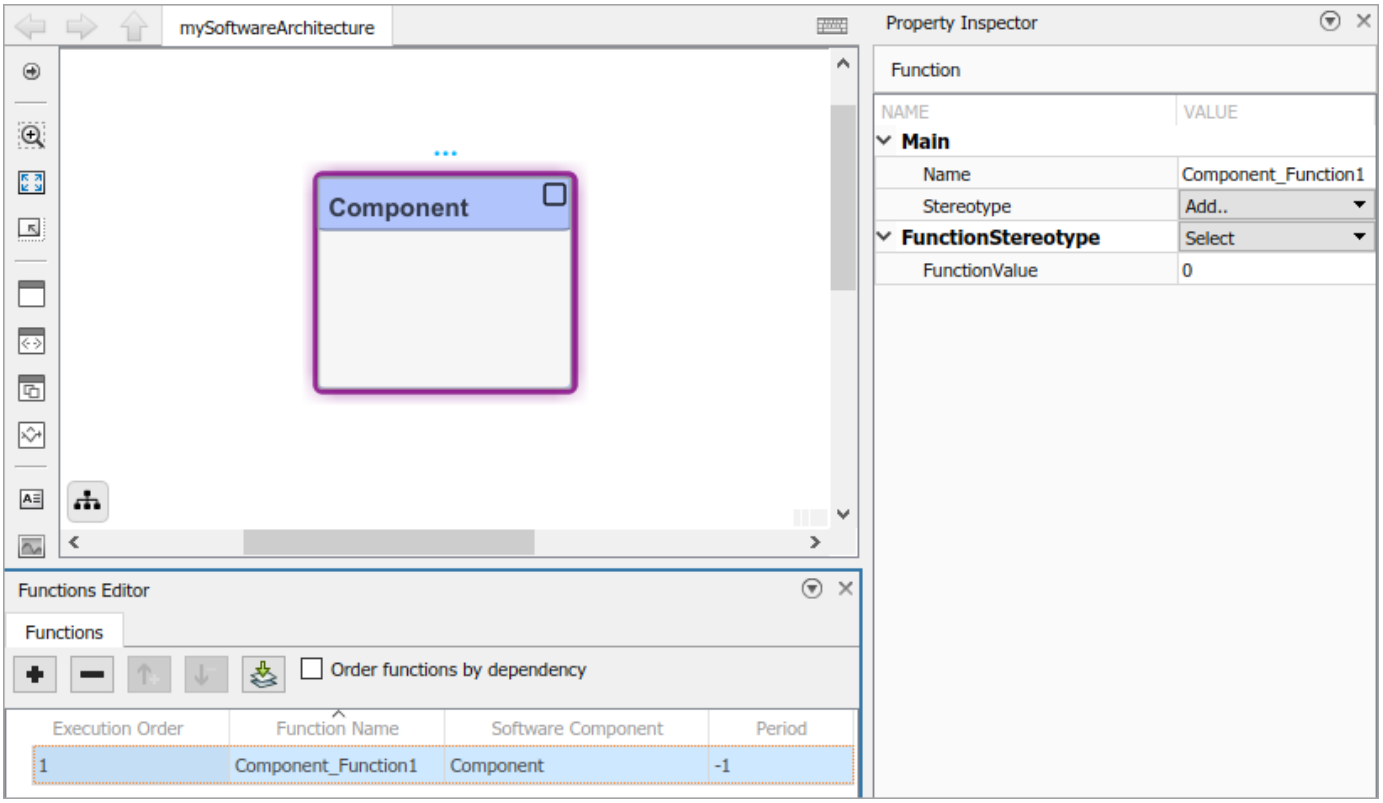
In R2022a, you can add stereotypes containing custom properties to software architecture functions.

First, define your function stereotypes using the **Profile Editor**.



Use the **Functions Editor** to select functions in your software component, apply stereotypes, view the stereotypes applied to your functions, and edit the stereotype property values.

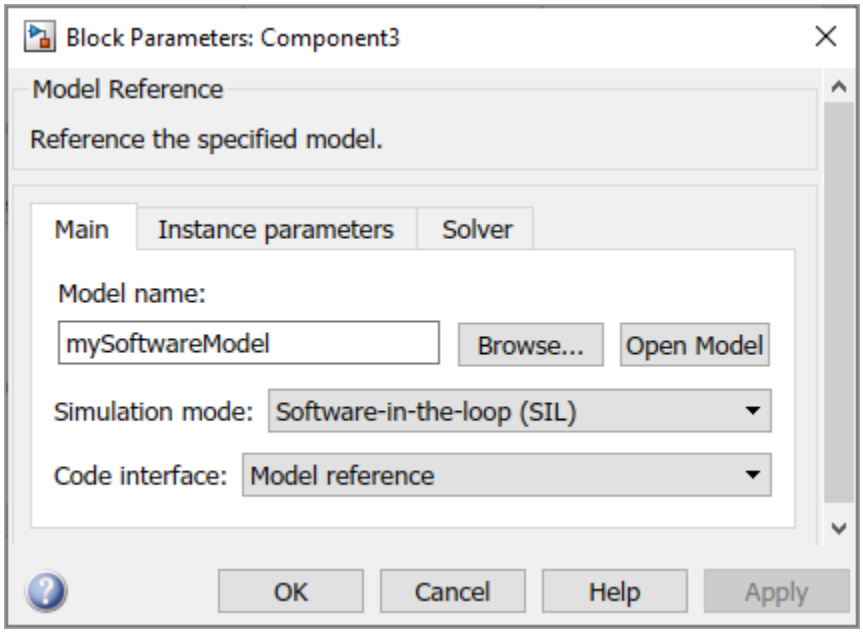
In this example, you can specify the value for the `FunctionValue` property of the stereotype called `FunctionStereotype`.



For more information, see Apply Stereotypes to Functions of Software Architectures.

Software-in-the-loop (SIL) and processor-in-the-loop (PIL) simulation support for reference components

In R2022a, for Reference Component blocks, you can change the block **Simulation mode** to `Software-in-the-loop (SIL)`, or `Processor-in-the-loop (PIL)`. Right-click the reference component and select `Block Parameters (Model Reference)` to access the block parameters.



Import and export functions of software architectures

In R2022a, you can import and export functions of your software architectures.

- Use `systemcomposer.exportModel` to output a `functions` field that contains a table with function information.
- Use `systemcomposer.importModel` to import a model with functions where the import structure can have a `functions` field that contains function information.

For more information, see Import and Export Functions of Software Architectures.

Author Model Advisor checks that run at edit-time

Starting in R2022a, if you have a Simulink Check license, you can author Model Advisor checks that run on architecture models. Because edit-time checks appear in the model canvas while you edit your model, they can help you catch issues earlier in the model design process. You can author edit-time checks that highlight issues on blocks and signals. To create a custom edit-time check, create a MATLAB class that derives from the `ModelAdvisor.EdittimeCheck` (Simulink Check) class. For an example, see Define Custom Edit-Time Checks that Fix Issues in Architecture Models (Simulink Check).

R2021b

Version: 2.1

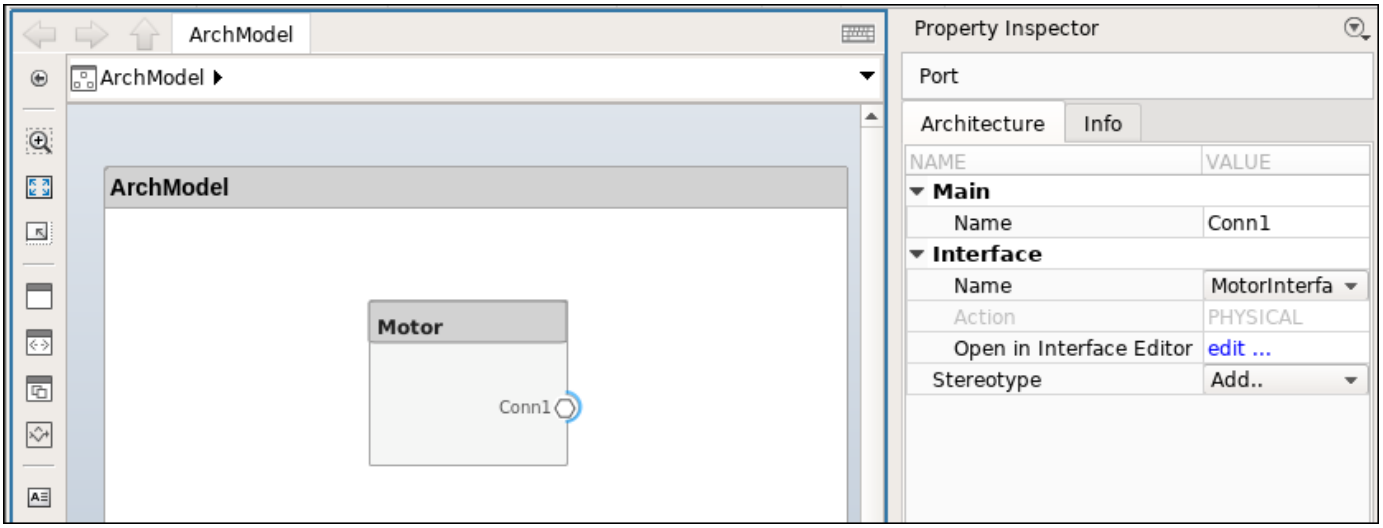
New Features

Bug Fixes

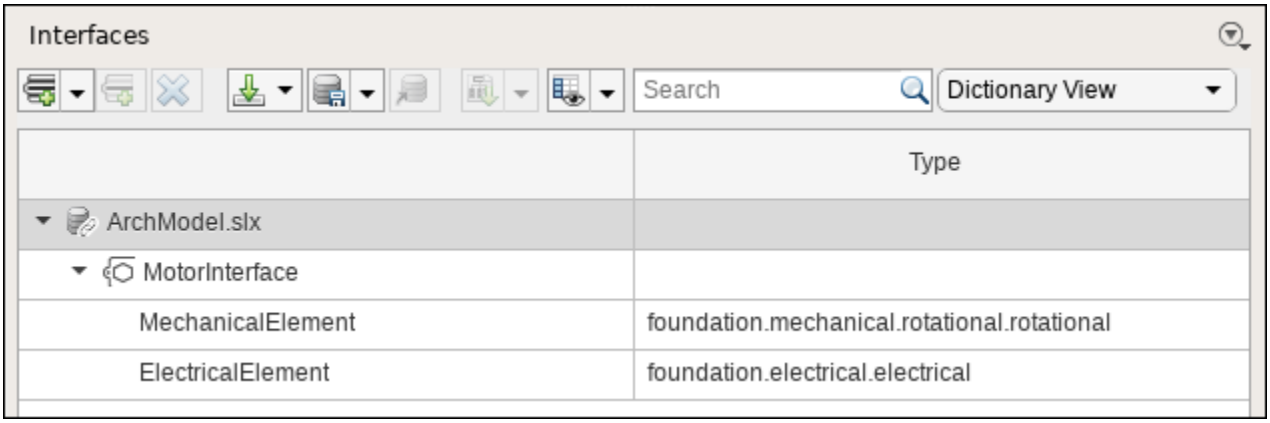
Compatibility Considerations

Physical Interfaces with Simscape: Create physical interfaces, ports, and connections on components

In R2021b, you can create physical interfaces, ports, and connections on components in architecture models and implement physical behaviors using Simscape. To implement physical behaviors, add Simulink subsystem behavior to a component.



Physical port interfaces can be aggregated from multiple Simscape domains.

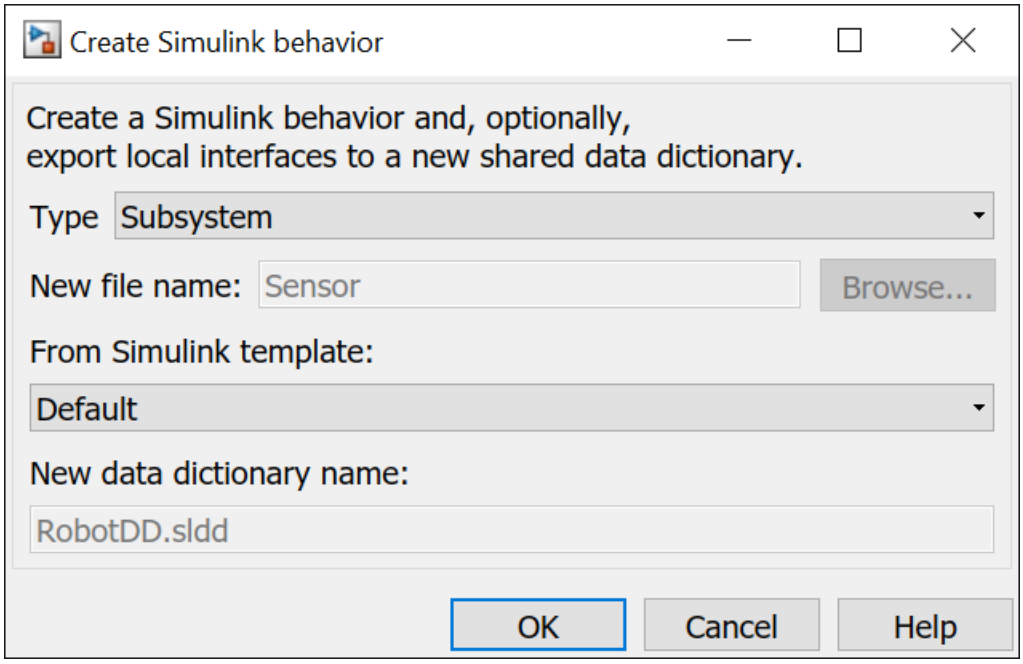


For more information, see Describe Component Behavior Using Simscape.

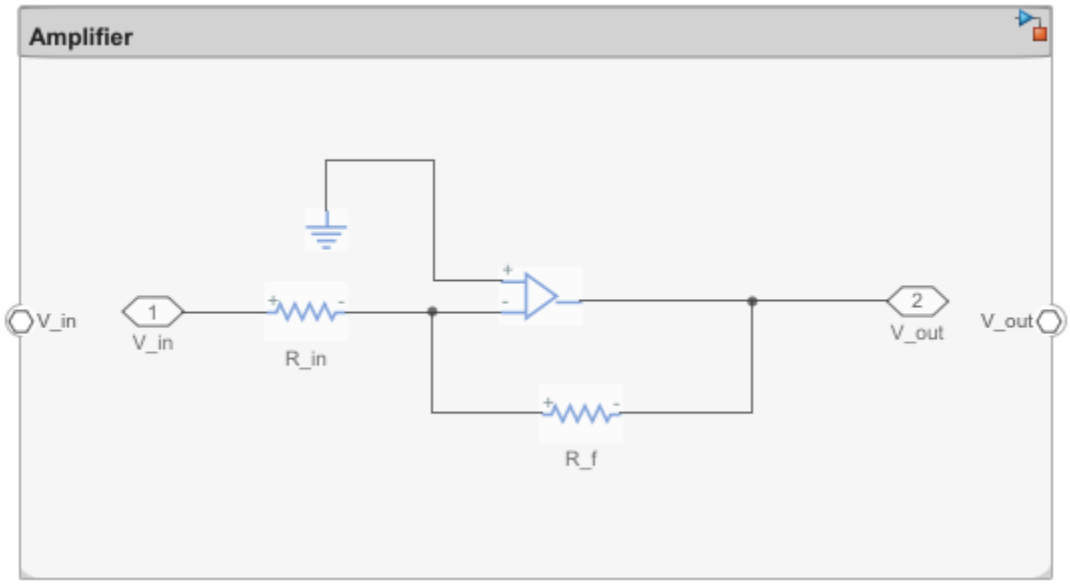
Simulink Subsystem Component: Add Simulink and Simscape behaviors to components

Create a Simulink subsystem that is part of the parent System Composer architecture model by adding Simulink subsystem behavior to a component.

In previous releases, Simulink behaviors were implemented as references to separate Simulink model files. In R2021b, Simulink behaviors can also be described within the same model as the architecture through subsystem components.



Subsystem components are required to author Simscape component behaviors with physical ports, connections, and blocks. For example, this amplifier physical system uses electrical domain blocks inside a subsystem component in a System Composer architecture model.



For more information, see Create Simulink Behavior Using Simulink Subsystem.

Create software architectures from existing components

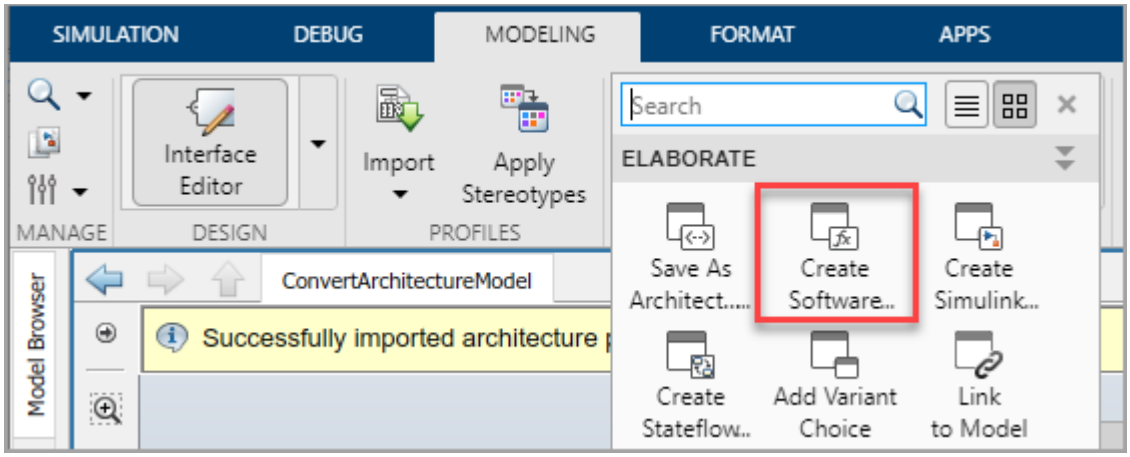
In R2021b, you can create a software architecture model from an existing component in System Composer.

To create a software architecture model from a component, you can use these two methods:

1 To create a software architecture model from the toolstrip:

   a Right-click the component and select `Create Software Architecture Model`.

   b Select the component, and, on the toolstrip, click **Create Software Architecture Model**.



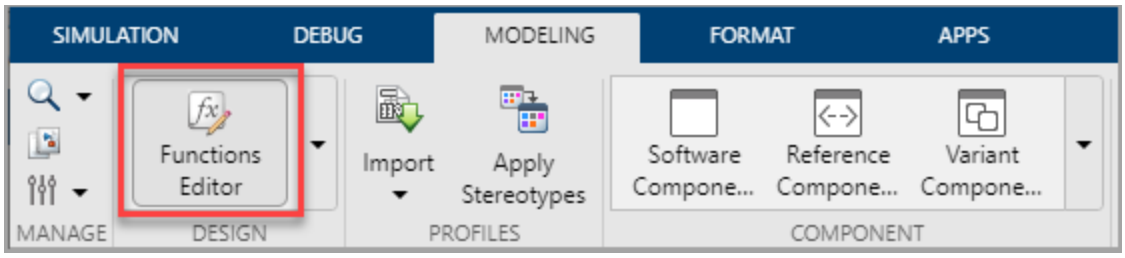
2 To create a software architecture programmatically, use the `createArchitectureModel` function.

For more information, see Create Software Architecture from Architecture Model Component.

Functions Editor: Visualize component functions in software architectures

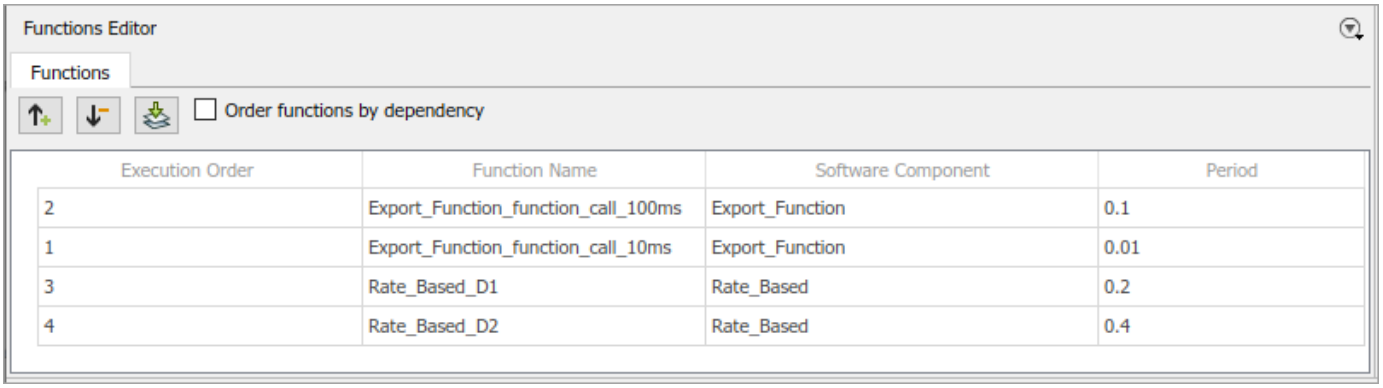
Use the Functions Editor to edit the simulation execution order and sample time of functions with inherited sample time (`-1`) in your software architecture.

Open the Functions Editor from the toolstrip.



Use the Functions Editor to:

- Update your software architecture model to automatically populate functions from any reference components.
- Arrange the execution order of the functions. Use the up and down arrows or drag and drop functions to sort them in the desired order.
- Edit sample times of the functions. Specify their period in the table.
- Order functions based on their data dependencies. Select **Order functions by dependency** check box.

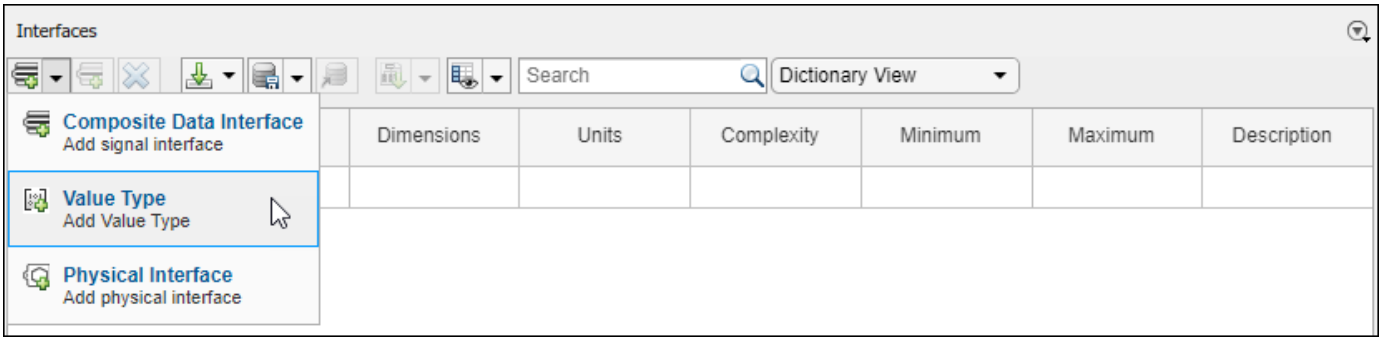


To get functions programmatically, use the new `systemcomposer.arch.Function` object.

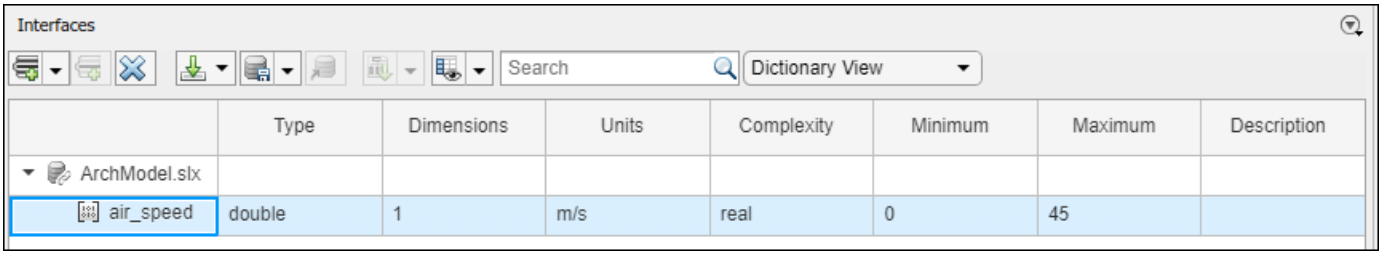
For more information, see Simulate and Deploy Software Architectures.

Value Types as Interfaces: Describe atomic pieces of data

In R2021b, you can apply a value type to a port to use it as a port interface. A value type is a description of an atomic piece of data. Data interfaces in System Composer are composite and structured with elements.



Using a value type as an interface means the interface has a top-level type, dimension, unit, complexity, minimum, maximum, and description.



You can also assign the type of data elements in data interfaces to value types and reuse the value types any number of times.

For more information, see Create Value Types as Interfaces.

For the new value type object, see `systemcomposer.ValueType`.

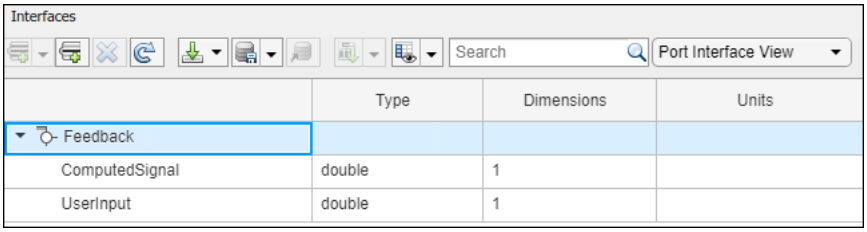
Compatibility Considerations

This table shows the new and removed programmatic interfaces in this release.

New	Removed	Rationale
`systemcomposer.interface.DataInterface`	`systemcomposer.interface.SignalInterface`	The class for a data interface has been renamed.
`systemcomposer.interface.DataElement`	`systemcomposer.interface.SignalElement`	The class for a data element has been renamed.

Owned Interfaces: Define interfaces local to ports

In R2021b, you can author locally defined interfaces that do not need to be reused, called owned interfaces. Owned interfaces are local to specific ports and are not shared in a data dictionary or the model dictionary. Owned interfaces represent a value type or data interface with attributes that describe a port.



For more information, see Define Owned Interfaces Local to Ports.

Compatibility Considerations

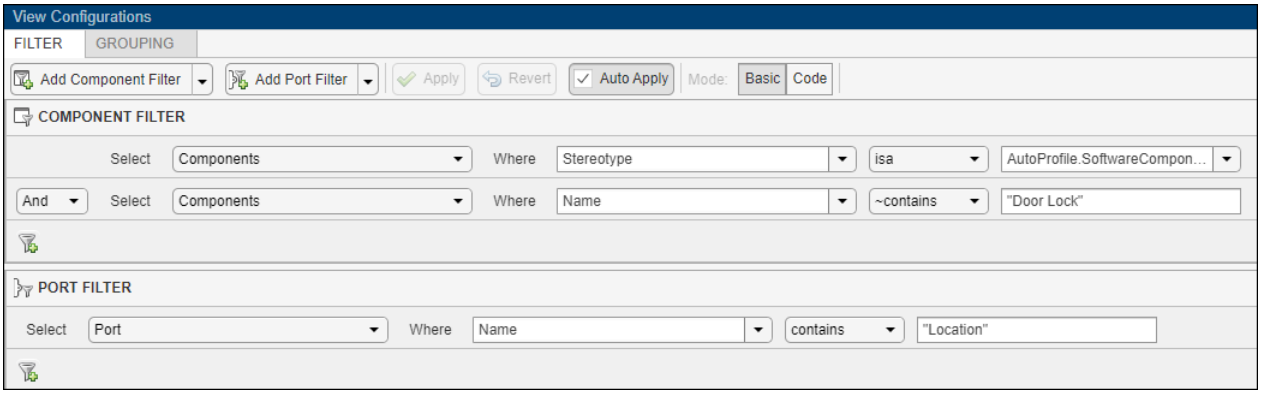
This table shows the new and removed programmatic interfaces in this release.

New	Removed	Rationale
`createInterface`	`createAnonymousInterface`	To better support owned interfaces, the term `"anonymous"` is being removed.

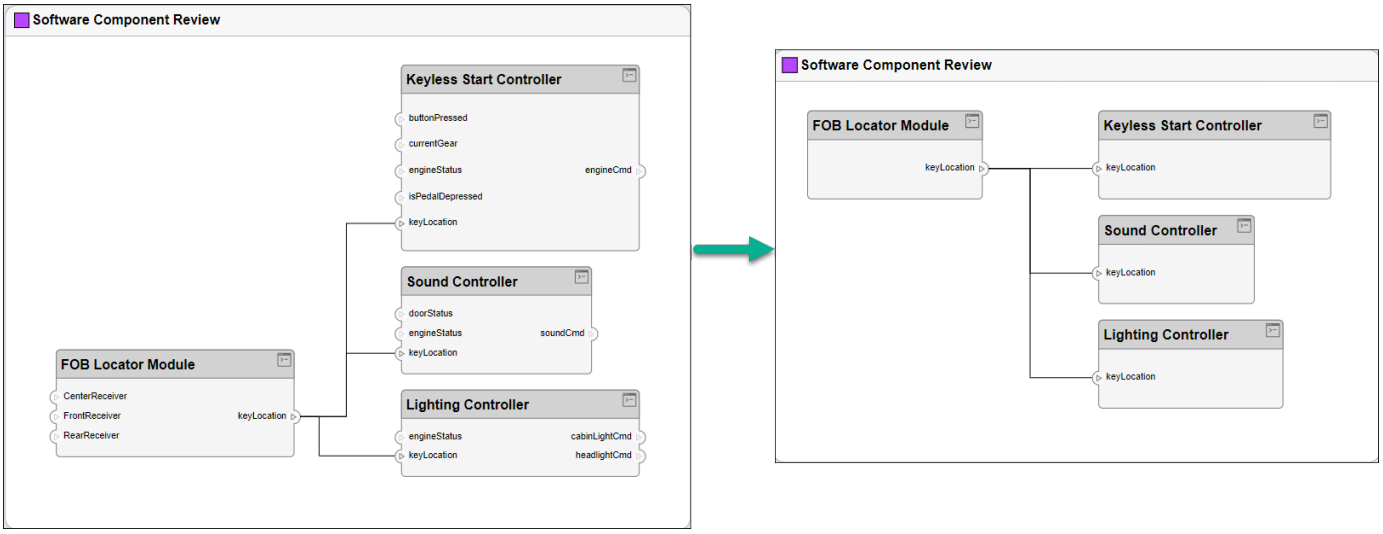
Architecture Views: Add port filters

In R2021b, you can filter ports in addition to components in architecture views using names, stereotypes, and property values.

The **View Configurations** panel now has an option to filter by a component filter or a port filter. You can also select the **Auto Apply** check box to automatically apply filters.



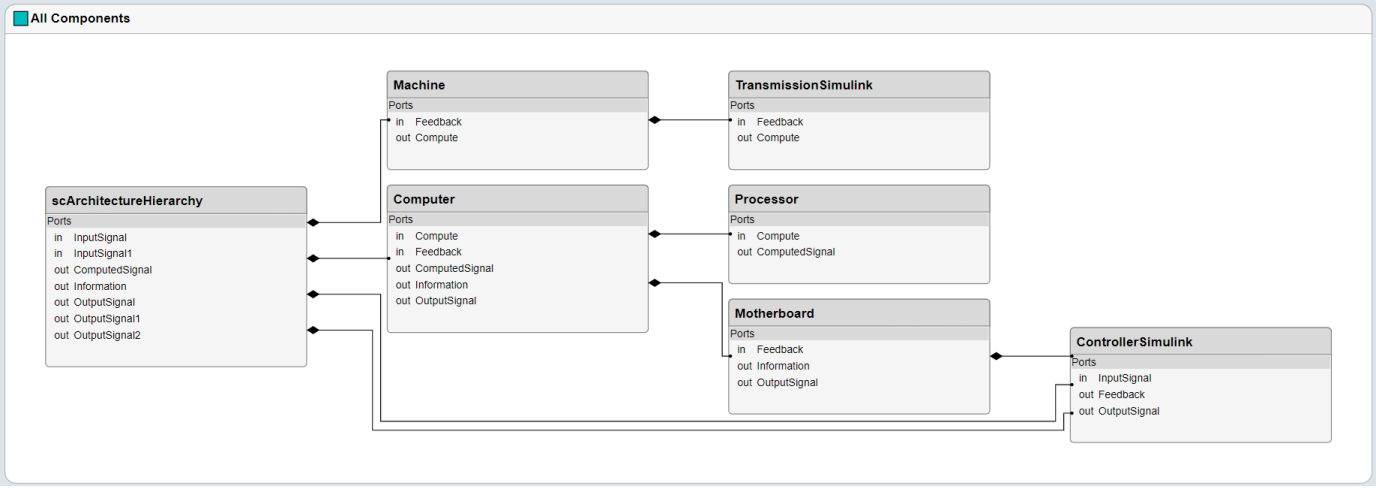
An architecture view is created using the additional query in the **Port Filter** box. The view is filtered to show all components with ports with `"Location"` in the name.



For more information, see Create Architecture Views Interactively.

Architecture Hierarchy: Display the hierarchy of unique component types

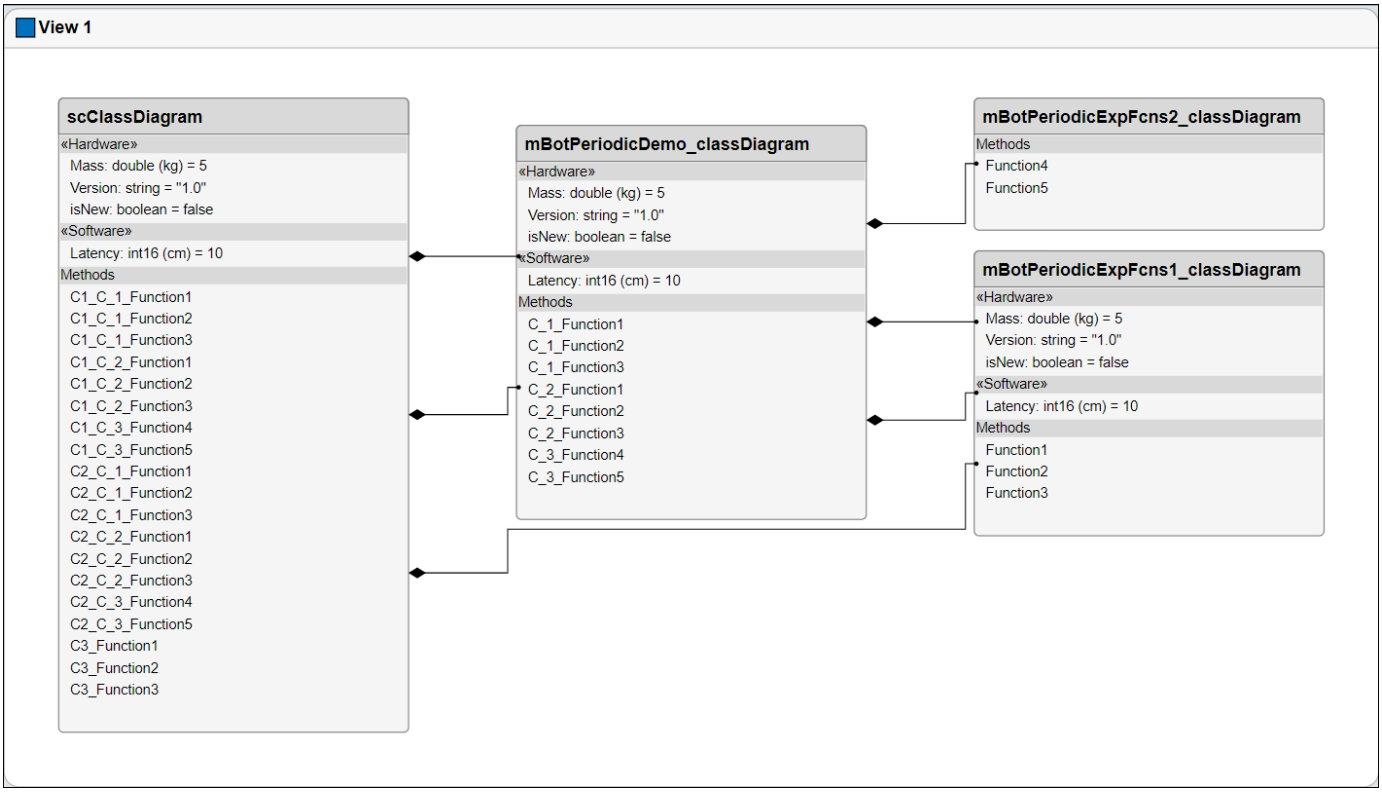
You can now visualize the hierarchy of architecture types of architecture models. Architecture hierarchy diagrams display unique component architecture types and their relationships using composition connections. In an architecture hierarchy view, each referenced model is represented only once.



For more information, see Display Component Hierarchy and Architecture Hierarchy Using Views.

Class Diagrams: Display software architecture model as a class diagram

You can now visualize a software architecture model as a class diagram. A class diagram is a graphical representation of a static structural model that displays unique architecture types of the software components optionally with software methods and properties. Class diagrams capture one instance of each referenced model and show relationships between them.



For more information, see Class Diagram View of Software Architectures.

Import and export software architectures

In R2021b, you can import and export software architectures.

- To import a software architecture to your model, use the `systemcomposer.importModel` function.

  ```
  archModel = systemcomposer.importModel(modelName,importStruct)
  ```

  If the `domain` field of `importStruct` is `"Software"`, the `importModel` function creates a new software architecture based on the structure of the MATLAB® tables.
- To export an existing software architecture, use the `systemcomposer.exportModel` function.

  ```
  exportedSet = systemcomposer.exportModel(modelName)
  ```

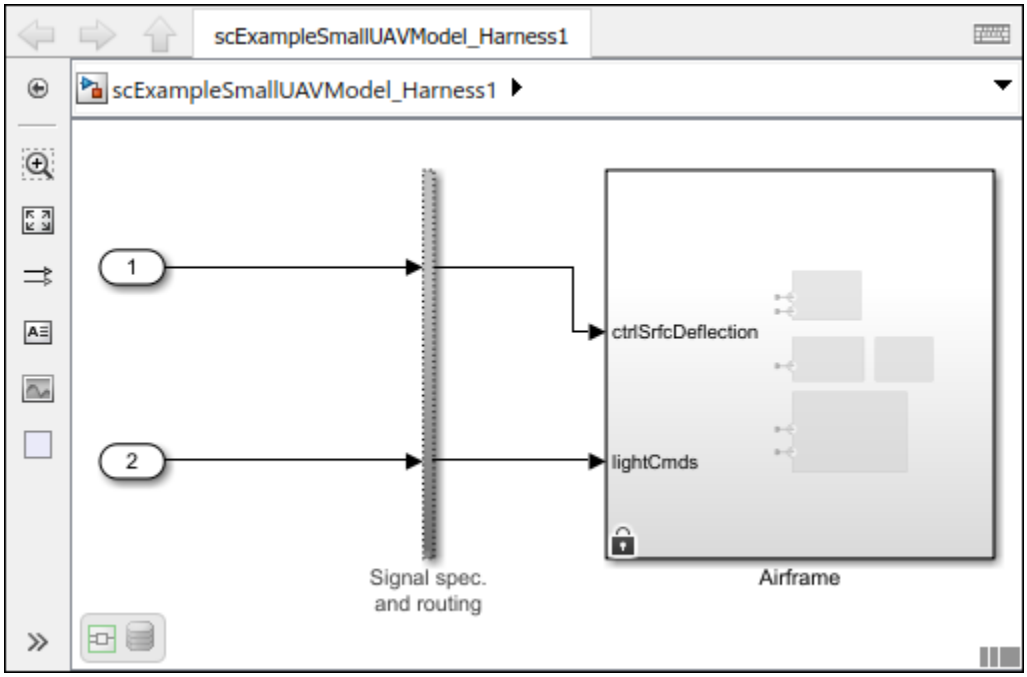
  The `exportModel` function returns a MATLAB structure with the following fields:

  - `components`, `ports`, `connections`, `portInterfaces`, and `requirementLinks` as MATLAB tables that capture components, ports, connections, port interfaces, and requirement links of the exported architecture.
  - `domain` with value `"Software"` to indicate that the exported architecture is a software architecture.

Test harnesses for System Composer components

In R2021b, you can create test harnesses for System Composer components in addition to Reference Component blocks.

A test harness enables you to isolate individual blocks for functional testing, behavior testing, and implementation-independent interface testing for requirements defined in System Composer using Simulink Requirements™. You can continue to manage your system interfaces inside the test harness and maintain synchronization with the System Composer model.



A Simulink Test™ license is required to create a test harness for a System Composer component to validate simulation results and verify design.

For more information, see Verify and Validate Requirements Using Test Harnesses on Components.

R2021a

Version: 2.0

New Features

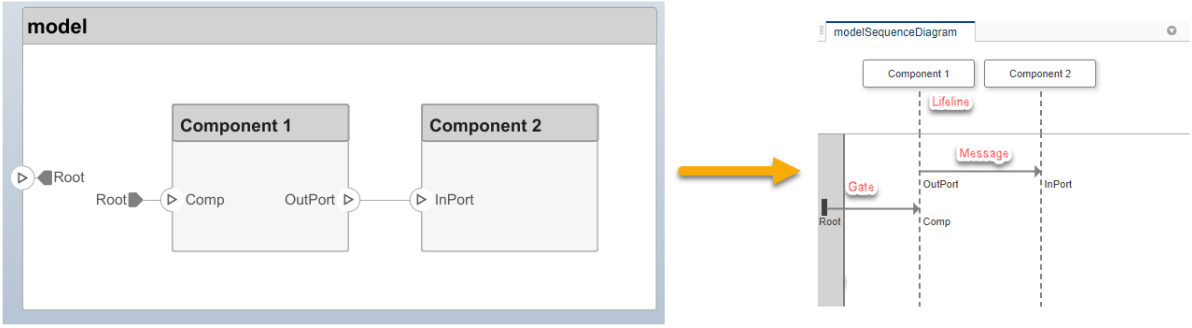
Bug Fixes

Compatibility Considerations

Sequence Diagrams: Describe system behavior as a sequence of interactions between components

You can use sequence diagrams in System Composer to describe the expected system behavior as a sequence of interactions between components of a system. You can create multiple sequence diagrams to represent different operational scenarios of the system. New lifelines or interactions authored on the sequence diagram are automatically reflected as new components or connections in the architecture model.

Sequence diagrams are available in the Architecture Views Gallery and support these features:



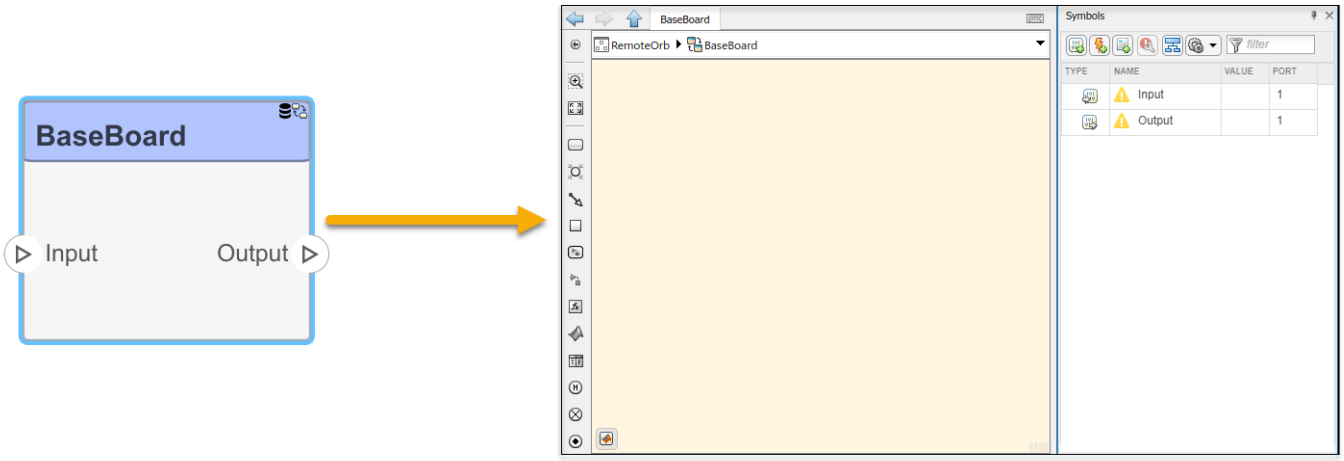
- Add lifelines to represent components and show what the component does in terms of events received and events sent out.
- Create child lifelines for child components.
- Add messages between lifelines that correspond to a connection between ports to show how components interact. These messages can have triggers and constraints attached to them.
- Create messages from root architecture ports to any lifeline component.
- Use the co-creation workflow. The software generates components and connections as you author new lifelines and messages in the sequence diagram.
- Make a marquee selection of messages and create a fragment. Click the **Add Operand** button to add operands for more complicated semantics like if-else branching or looping.
- When a mismatch occurs from the architecture model the sequence diagram references, click the **Check Consistency** button to highlight lifelines pointing to missing components for manual reconciliation.

For more information, see Define Sequence Diagrams.

State Charts: Describe component behavior using Stateflow charts to represent modes of operation

In R2021a, you can add Stateflow® chart behavior to a component in a System Composer architecture model.

- Add Stateflow chart behavior to a component.



- The new Stateflow chart behavior for a component is embedded within the same `.slx` file as the parent architecture model, reducing the need for multiple model files.

For more information, see Add Stateflow Chart Behavior to Architecture Component.

Software Architecture: Simulate and deploy software functions from System Composer

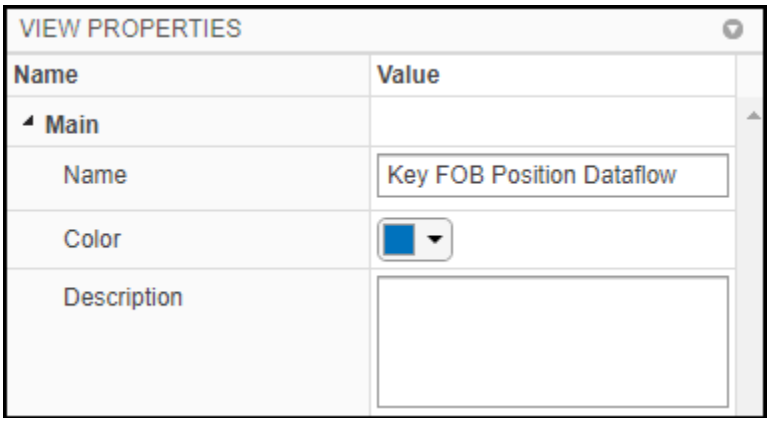
Use System Composer to author software architectures composed of software components, ports, and interfaces. Design your software architecture model, define the execution order of functions from your components, simulate your design at the architecture level, and generate production code. For more information, see Author Software Architectures.

For an example that shows how to author the software architecture of a throttle position control system in System Composer, see Modeling the Software Architecture of a Throttle Position Control System.

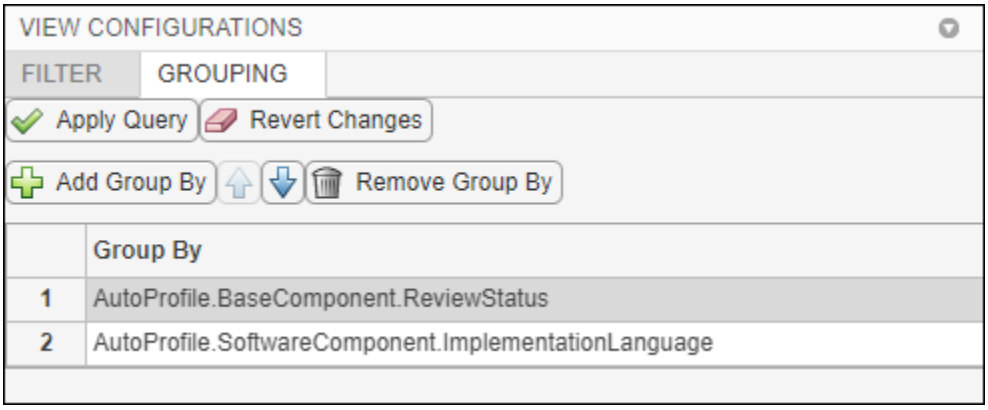
Updates to the Architecture Views Gallery and Programmatic Interfaces

The Architecture Views Gallery updates include UI improvements and new programmatic interfaces for views.

- Use View Properties on the right of the views gallery when a view is selected. Properties include Name, Color, and Description.



- Create and define the **Filter** and **Grouping** for views in **View Configurations** at the bottom of the views gallery.
- **Group By** multiple property values.



- New views programmatic interfaces are available to streamline creating views programmatically. You can now add elements to the view independent of ordering and to remove elements cleanly. Existing programmatic interfaces are removed.

For more information, see Create Architecture Views Interactively and Create Architectural Views Programmatically.

Compatibility Considerations

This table shows the new and removed programmatic interfaces in this release.

New	Removed	Rationale
`createView`	`createViewArchitecture`	The new object `systemcomposer.view.View` is created with a name and optionally with a query, grouping criteria, and the option whether to include references. The `Root` property of a `View` object is a `systemcomposer.view.ElementGroup` object.
`addElement`	`addComponent`	The `addElement` method is used to populate the view manually and exists under the `systemcomposer.view.ElementGroup` object.
`removeElement`	`removeComponent`	The `removeElement` method is used to depopulate the view manually and exists under the `systemcomposer.view.ElementGroup` object.
`createSubGroup`	`createViewComponent`	The `createSubGroup` method is used to add subgroups under the main `systemcomposer.view.ElementGroup` object.

Performance and Scalability Improvements

In R2021a, there are performance and scalability improvements for System Composer features.

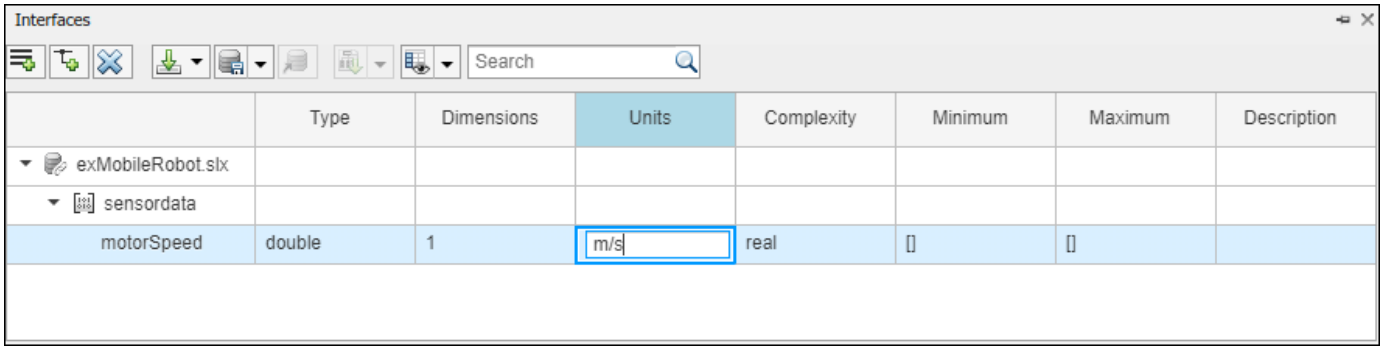
- Model-building programmatic interfaces including `connect`, `addPort`, `addComponent`, and `addVariantComponent` now perform operations faster.
- Queries that query for model elements with specific stereotypes and model elements that satisfy constraints on stereotype property values now perform better. For examples, see Find Elements in a Model Using Queries.
- Interface editor scalability enhancements include loading improvements and an improved user experience on interface search, sort, expand, and collapse.

Interface Editor enhancements

In R2021a, the Interface Editor features three enhancements.

- Use multiple columns to track interface element properties. Column names include: Type, Dimensions, Units, Complexity, Minimum, Maximum, and Description.

- Show and hide columns by toggling the button.
- After creating interface elements, you can edit the properties in the columns from the Interface Editor.



For more information, see Define Interfaces.

R2020b

Version: 1.3

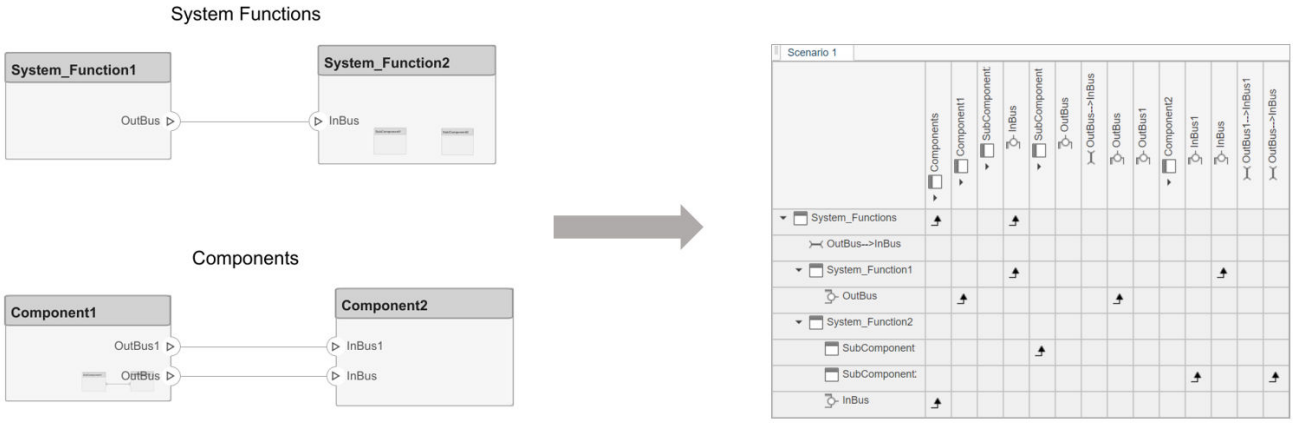
New Features

Bug Fixes

Model to Model Allocations: Establish traceable and directed relationships between architectural elements in source and target models

System Composer allocations allow you to establish a direct relationship from architectural elements in one model such as components, ports, and connectors, to architectural elements in another model. For more information, see Create and Manage Allocations.

You can use allocations to establish relationships from software components to hardware components and to indicate the deployment strategy. You can allocate different instances of components, ports, and connectors and use the allocation to perform various analyses such as resource-based allocation analysis. For an example, see Allocate Architectures in a Tire Pressure Monitoring System.



An allocation set is a collection of allocation relationships between two models. The allocation set is stored with the `.mldatx` file extension. You can create an allocation set in the Allocation Editor. Upon the allocation set creation, a matrix appears where the source model elements are the rows and the target model elements are the columns. You can open the allocation editor from the **Allocation Editor** button in the toolstrip or by typing the `editor` command in the MATLAB Command Window.

Enhanced workflows with Architecture Views

In R2020b, workflows for authoring and generating multiple views are enhanced to simplify view editing and filtering. There are six enhancements:

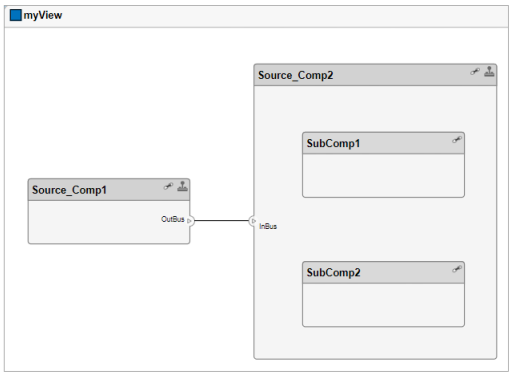
- You can start editing a view after selecting the view in the **View Browser** pane.
- You can add components from the composition tree in the **Model Components** pane by dragging and dropping them from the composition tree.
- You can use the keyboard shortcut **Delete** to delete components from the view.
- You can use the **Group** button in the toolstrip to create a view component based on the selected components.
- You can ungroup components by using the **Ungroup** button in the toolstrip.
- View filtering is now placed in a separate pane in which you can customize the query and apply it to your view.

For more information, see Create Architecture Views Interactively.

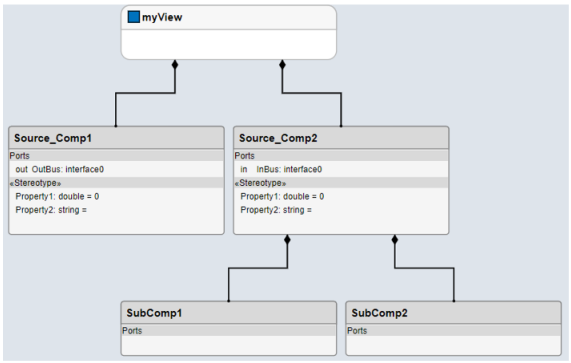
Hierarchy Views: Display component hierarchy as a tree diagram

The Hierarchy View allows you to visualize component hierarchy together with component stereotypes, stereotype properties, and the reference type a component instantiates. Hierarchy diagrams show the same set of components visible in the component diagram view, and components are selected and filtered in the same way. Any component diagram view can be optionally represented as a hierarchy diagram. The Hierarchy View flattens the component diagram view and displays in a tree form. You can see the Hierarchy View by selecting **Hierarchy diagram** in the Architecture Views Editor.

Component diagram



Hierarchy diagram



The hierarchy diagrams show a single root, which is the view specification. The root corresponds to the containing system box shown in the component view. The lines between components originate from the parent component with the filled diamond symbols that are used in UML compositions and end with a terminating dot at the child component.

For more information, see Display Component Hierarchy Using Hierarchy Views.

Referenced Data Dictionaries: Organize interfaces in a hierarchy of data dictionaries

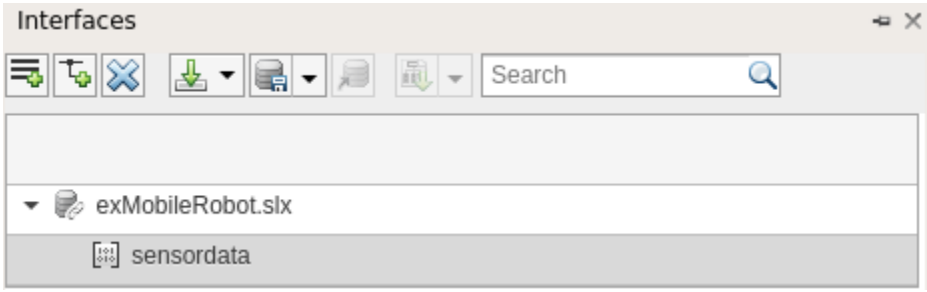
In R2020b, an interface data dictionary can reference other data dictionaries, creating a hierarchy in which the dictionary aggregates interfaces defined in its referenced dictionaries. A System Composer model linked to the root (top-level) dictionary of a hierarchy can access and use interfaces defined in the root as well as any of the dictionaries it references directly or indirectly.

For more information, see Save, Link, and Delete Interfaces.

Interface Editor enhancements

In R2020b, there are three enhancements to the Interface Editor.

- The **Interfaces** pane is now placed at the bottom of the model window.

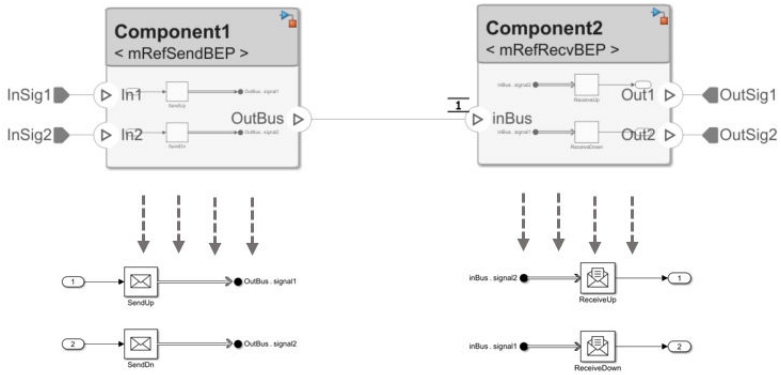


- You can search the interface table on the first column.
- You can sort interfaces and specify the sorting criteria.

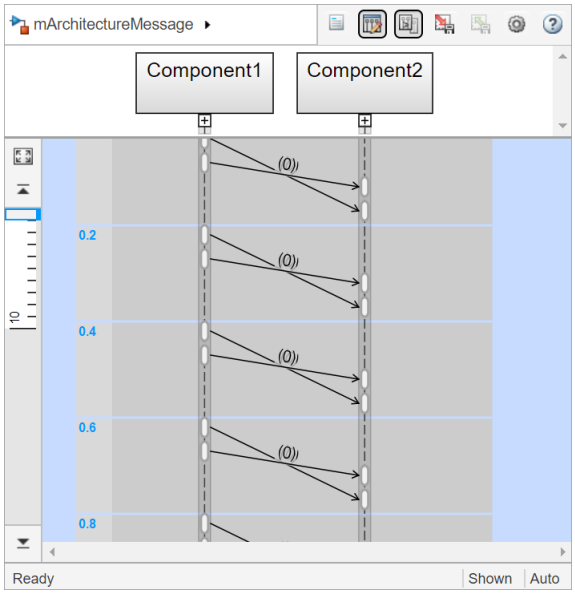
For more information, see Define Interfaces.

Support for component behavior using Simulink models with message input and output

You can define component behavior by linking a Simulink behavior model with message input and output. For more information about Simulink messages, see Simulink Messages Overview (Simulink).



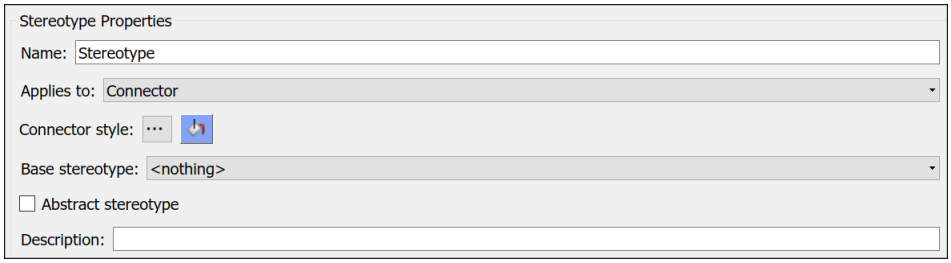
Use the **Sequence Viewer** tool to visualize events that represent the flow of messages, functions calls, and state transitions. To activate event logging, on the **Simulation** tab, in the **Prepare** section, select **Log Events**. To visualize the logged events, go to the **Review Results** section and select the **Sequence Viewer**.



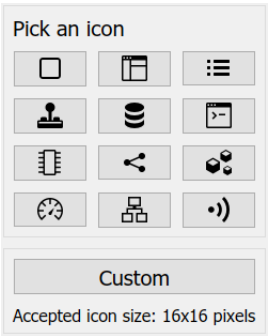
Stereotype-Based Styling: Style connectors based on stereotypes and use custom icons for component stereotypes

In R2020b, there are two stereotype-based styling enhancements in the Profile Editor:

- You can style architecture connectors using the stereotype settings. You can style connectors by using connector, port, or port interface stereotypes. Customize styling provides various color and line style choices. Connector styles are also reflected in architecture and spotlight views.



- You can use custom icon images for component stereotypes. Custom icons support `.png`, `.jpeg`, or `.svg` image files of size 16-by-16 pixels. The custom icons are displayed as badges on the components for which the stereotypes are applied.



For more information, see Define Profiles and Stereotypes.

Support for protected models as component behaviors

In R2020b, you can link a component in your architecture model to a protected Simulink model to create component behaviors. You can convert an already linked Simulink behavior model to a protected model, and this change is reflected after refreshing the model.

For more information, see Implement Components in Simulink.

Import and export requirement links along with the architecture models

In R2020b, you can:

- Export your architecture models with requirement links to your system requirements.

  When you export your architecture model using the `exportModel` function, the output structure has a `requirementLinks` field that contains all the link information present in the model.
- Import links to requirements and reconstruct them in your System Composer model using `importModel`.

  You can import requirement links to your model by using the new `requirementLinks` argument in `systemcomposer.importModel(modelName,components,ports,connections,interfaces,requirementLinks)`.

  For more information, see Import and Export Architecture Models.

R2020a

Version: 1.2

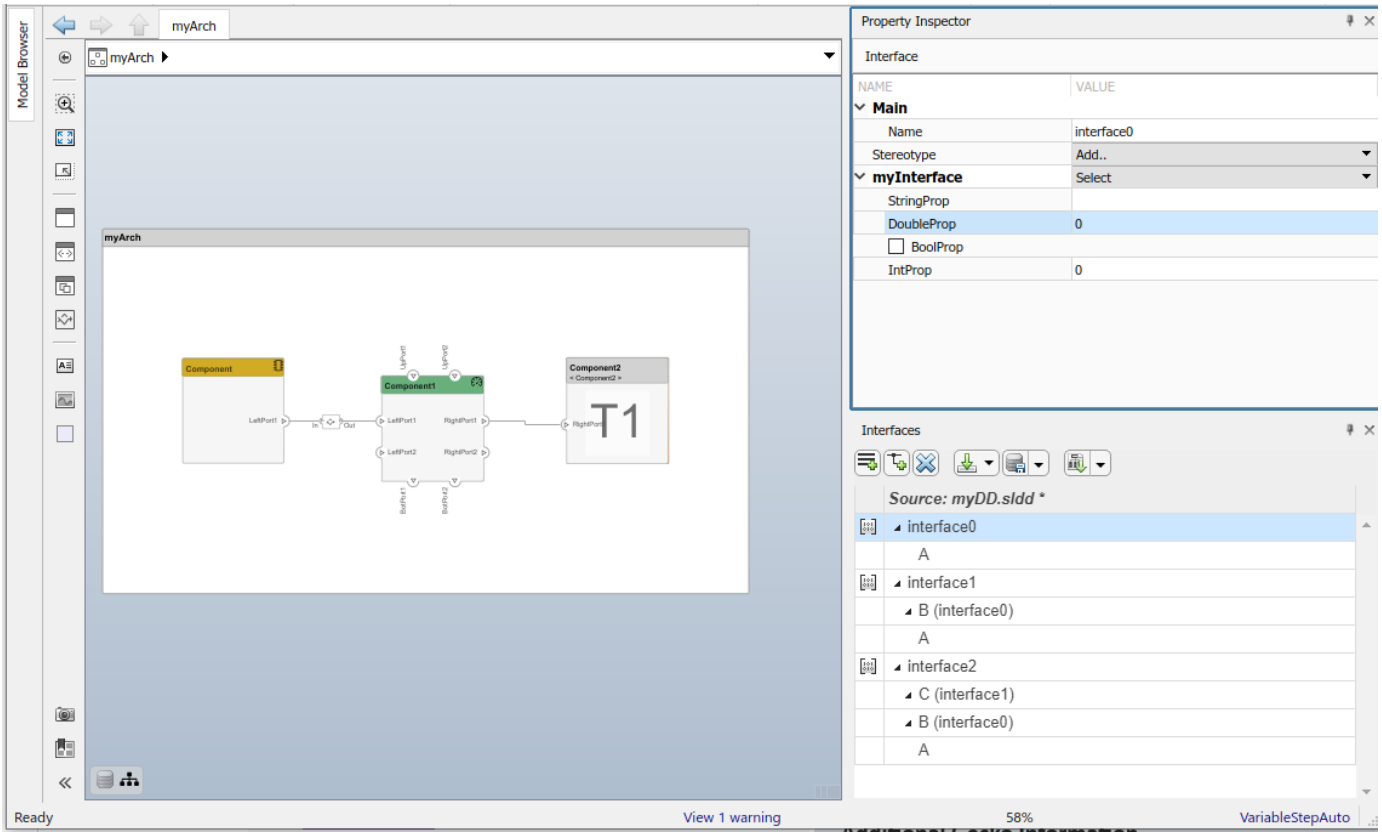
New Features

Stereotype-Based Styling: Associate a color with component stereotypes

Profiles and stereotypes are used to apply custom metadata on the architecture model elements. Element styling is an additional visual cue that indicates applied stereotypes. Use a preconfigured set of color options for component stereotypes to style the architecture component headers.

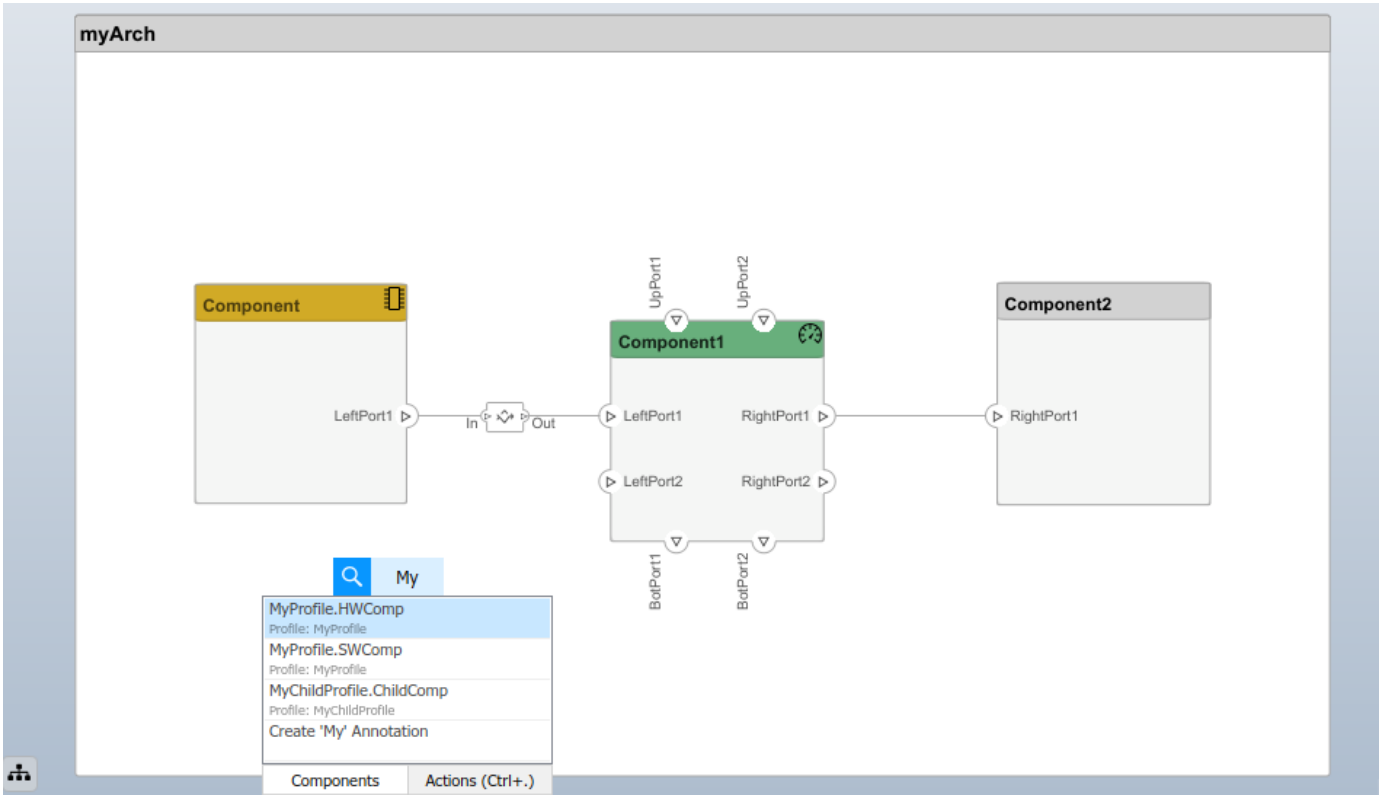
Interface Stereotypes: Apply stereotypes for custom metadata on interface objects

In R2020a, you can add stereotypes containing custom properties to port interfaces. Architecture profiles can be imported into shared dictionaries to create and manage stereotyped interfaces that are shared across multiple models, which enables collaborative workflows. Specify stereotypes that apply to interfaces in the Profile Editor. You can view and edit stereotype properties in the Property Inspector.



Quick Insert Stereotypes: Insert stereotyped components directly by typing on the canvas

You can now insert components preconfigured with all available component stereotypes in the profiles applied to a model using the quick insert capability of the System Composer canvas, which is enabled by typing into an empty location in the canvas.

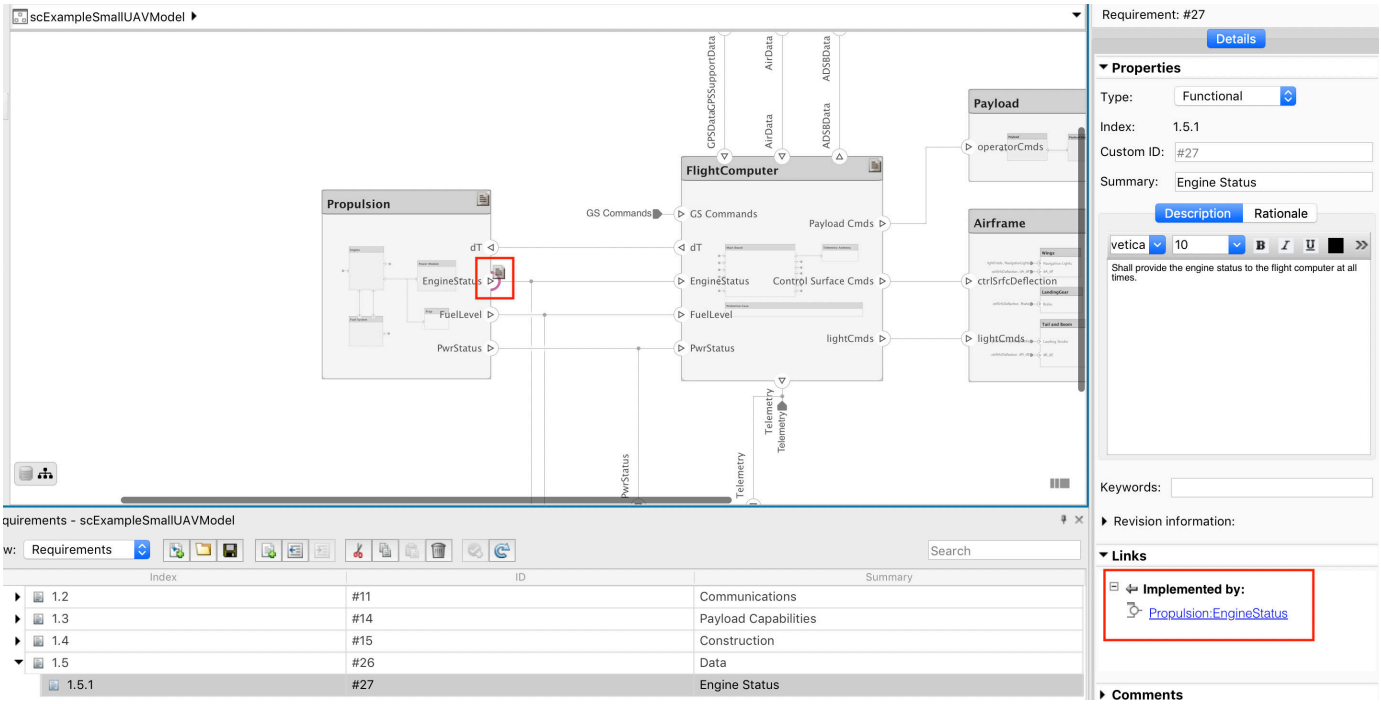


Model Templates for Referenced Models: Select from preconfigured templates when creating new referenced architecture or behavior models

To maintain specialized settings when creating Simulink behaviors or new architecture models linked to components, you can choose to use an existing template model from the palette of available template models on the Simulink start page. Specialized settings can include configuration parameters such as the solver type, diagnostic settings, and code generation settings. Use the template to specify copyright annotations and organizational styles.

Requirements on Component Ports: Link requirements to component ports

In R2020a, you can link requirements to ports in a System Composer model using the Requirements Perspective. Create requirement sets, organize requirements into hierarchies, and link requirements to components and ports using Simulink Requirements. Annotate architecture models with requirement information and navigate to the source requirement.



Partial Architecture Model Load: Improved loading for large models

System Composer now loads the architecture part of the model without completely loading the underlying block diagrams of models that are linked to components. For example, opening an architecture model with hundreds of reusable components and behaviors no longer loads every block diagram in the model hierarchy. Models are fully loaded when System Composer workflows need them or you choose to open them explicitly.

R2019b

Version: 1.1

New Features

Bug Fixes

Architecture Views

You can author and generate multiple views for specific viewpoints for design, analysis or communication. Create a view by writing queries against the model using the System Composer model query language. See Creating Architecture Views Interactively.

Interface Adapter

Visually specify an interface mapping between components with different but compatible interfaces. For example, map a signal with different interface names. See Interface Adapter.

Import and Export Architecture Models

Import an architecture into System Composer using Microsoft® Excel® spreadsheets or MATLAB tables. The tables define components, ports, and connections. See Importing and Exporting Architecture Models.

AUTOSAR Software Architecture

AUTOSAR Composition Editor: Author compositions, simulate functional behavior with basic software services using Composition Editor.

Support for architecture and composition modeling for the AUTOSAR Classic Platform. You can:

- Create AUTOSAR architecture models.
- Use AUTOSAR composition editor to add and connect Composition and Component blocks.
- Link AUTOSAR components to Simulink Requirements.
- Define AUTOSAR component behavior by creating or linking Simulink implementation models.
- Configure AUTOSAR scheduling and simulation, including AUTOSAR Basic Software services.
- Generate and package composition AUTOSAR XML description and component code.

R2019a

Version: 1.0

New Features

Introducing System Composer

System Composer enables the definition, analysis, and specification of architectures and compositions for model-based systems engineering and software design. With System Composer, you allocate requirements while refining an architecture model that can then be designed and simulated in Simulink.

System Composer lets you create architecture models that describe a system in terms of components and interfaces. You can also populate an architecture model from the architectural elements of Simulink designs or C/C++ code. You can create custom live views of the model to study specific design or analysis concerns. With these architecture models, you can analyze requirements, capture properties via stereotyping, perform trade studies, and produce specifications and ICDs.

Composition Editor

Author and edit architecture models in the Composition Editor. Model physical and logical architecture of a system. Create a visual representation of components, ports, and connections and specify information exchange between components with interfaces. Decompose components to add detail and define hierarchical relationships. Create reusable architectures.

Spotlight Views

You can create Spotlight views to analyze component dependencies and hierarchy. A Spotlight view is a simplified view of a model that captures the upstream and downstream dependencies of a specific component of interest. You can shift the spotlight to a different component from within the Spotlight view. You can also trace an element from the Spotlight view back to the composition.

Linking Simulink Behavior Models

You can use Simulink models with System Composer to define component behavior by creating or linking to a Simulink behavior model. Take advantage of System Composer architecture editing and analysis capabilities for Simulink behavior models by exporting them as architecture models.

Linking and Managing Requirements

You can link components to requirements using the Requirements perspective. Create requirement sets, organize requirements into hierarchies, and link requirements to components using Simulink Requirements. Annotate architecture models with requirements information and navigate to the source requirement.

Stereotypes and Profiles

You can create stereotypes as extensions of components, ports, and connections by defining additional properties. Use Profile Editor to define profiles as self-consistent sets of stereotypes. Import profiles into architecture models. Assign stereotypes to model elements.

Architecture Model Analysis

You can use MATLAB analytics with System Composer programmatic interfaces to write scripts to generate data that can be used for trade studies or for verifying nonfunctional requirements.