

System Composer™

User's Guide



MATLAB® & SIMULINK®

R2019a



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™ User's Guide

© COPYRIGHT 2019 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.

Revision History

March 2019 Online only New for Version 1.0 (Release 2019a)

1 **Architecture Model Editing**

| | |
|---|-------------|
| Compose Architecture Visually | 1-2 |
| Create an Architecture Model | 1-2 |
| Components | 1-5 |
| Ports | 1-10 |
| Connections | 1-13 |
| Importing Architectures | 1-17 |
| | |
| Decompose and Reuse Components | 1-18 |
| Decompose a Component | 1-18 |
| Create a Reference Architecture | 1-20 |
| Use a Reference Architecture | 1-22 |
| Inline a Reference Architecture | 1-23 |
| Create Variants | 1-25 |
| | |
| Create Spotlight Views | 1-28 |

2 **Requirements**

| | |
|--|------------|
| Link and Trace Requirements | 2-2 |
| | |
| Manage Requirements | 2-9 |

Interface Management

3

| | |
|---|------|
| Define Interfaces | 3-2 |
| Create Interface | 3-2 |
| Assign Interfaces to Ports | 3-7 |
| Save and Link Interfaces | 3-11 |

Define Architectural Properties

4

| | |
|--|------|
| Define Profiles and Stereotypes | 4-2 |
| Create a Profile and Add Stereotypes | 4-3 |
| Add Properties with Stereotypes | 4-4 |
| Use Stereotypes and Profiles | 4-8 |
| Apply a Stereotype | 4-8 |
| Remove a Stereotype | 4-11 |
| Extend a Stereotype | 4-12 |

Use Simulink Models with System Composer

5

| | |
|---|-----|
| Implement Components in Simulink | 5-2 |
| Create a Simulink Behavior Model | 5-2 |
| Link to an Existing Simulink Behavior Model | 5-4 |
| Extract Architecture from Simulink Model | 5-6 |

6

Analyze Architecture **6-2**
 Set Tags and Properties for Analysis **6-2**
 Create a Model Instance for Analysis **6-4**
 Write Analysis Function **6-6**
 Run Analysis Function **6-8**

Architecture Model Editing

- “Compose Architecture Visually” on page 1-2
- “Decompose and Reuse Components” on page 1-18
- “Create Spotlight Views” on page 1-28

Compose Architecture Visually

| In this section... |
|--|
| "Create an Architecture Model" on page 1-2 |
| "Components" on page 1-5 |
| "Ports" on page 1-10 |
| "Connections" on page 1-13 |
| "Importing Architectures" on page 1-17 |

Create and edit visual diagrams to represent system architecture in System Composer™. Use visual architecture elements: components, ports, and connections. Model hierarchy in architecture by decomposing components. Navigate through the hierarchy.

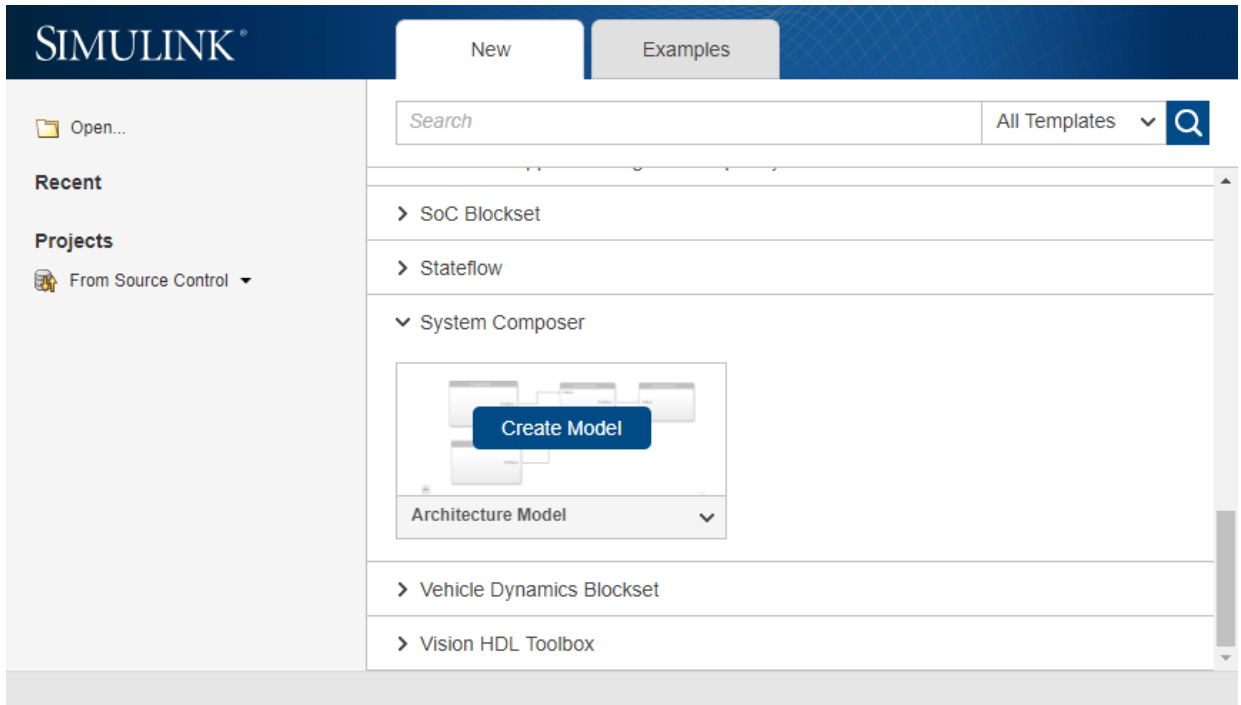
Create an Architecture Model

Start with a blank architecture model to model physical and logical architecture of a system. Use one of these methods to create an architecture model:

- From the Simulink start page — Type:

```
>> systemcomposer
```

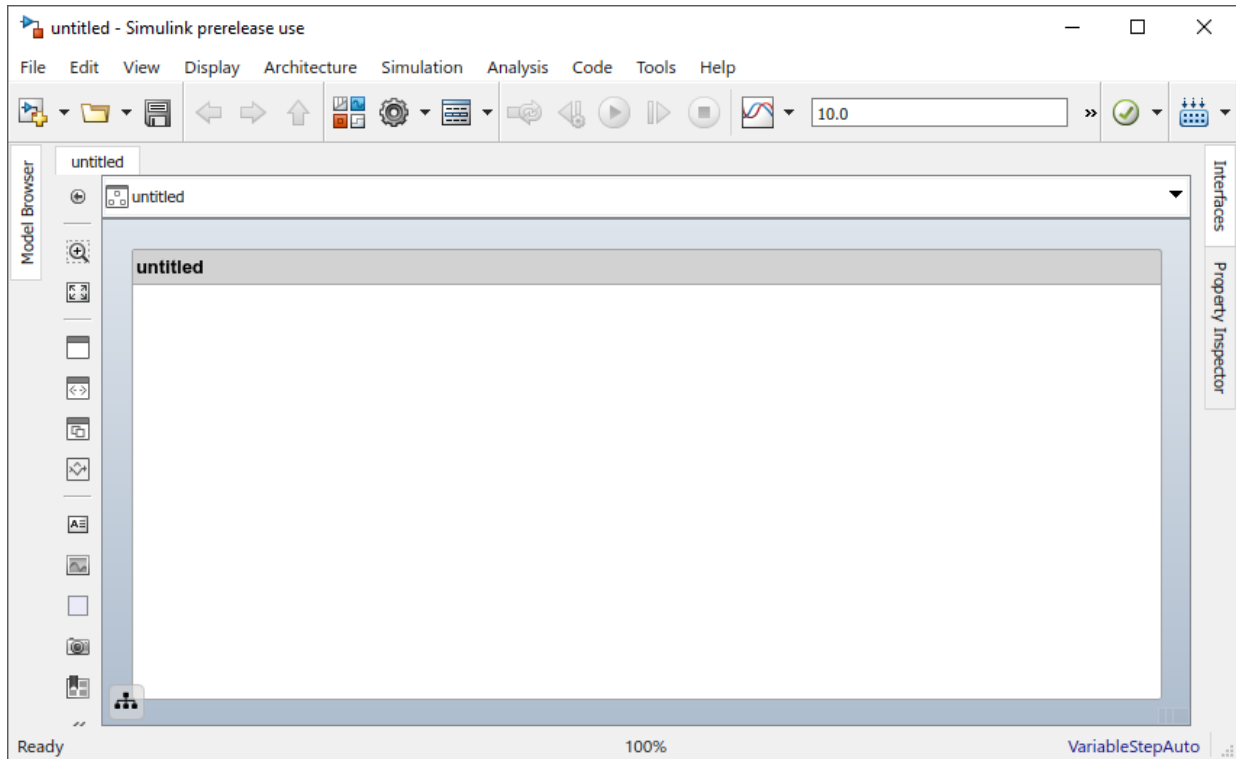
at the command line and select **Architecture Model** under the System Composer tab.



- From a Simulink model or a System Composer architecture model — Select **File > New > Architecture**.
- From the MATLAB command line — Type:

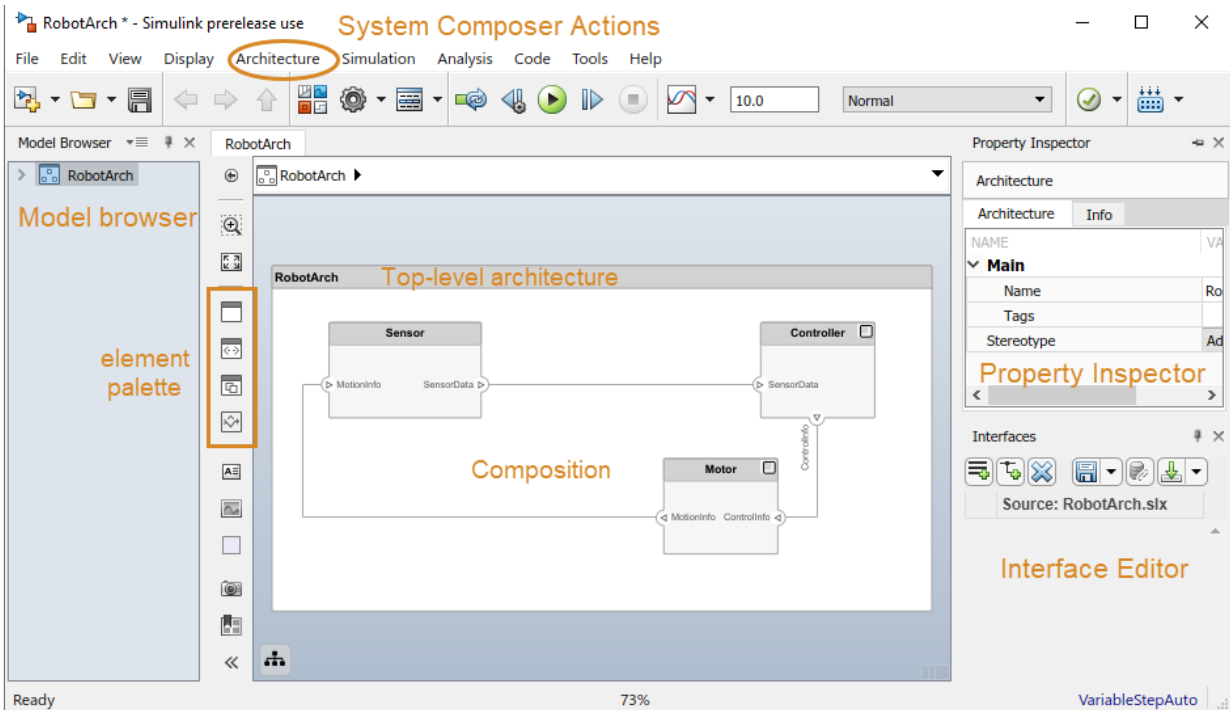
```
archModel = new_system('ModelName','Architecture');  
open_system(archModel)
```

at the command line, where `ModelName` is the name of the new model.



Save the architecture model at any time using **File > Save** or **File > Save As**. The architecture model is saved as an `.slx` file.

The architecture model includes a top-level architecture that holds the composition of the system. This top-level architecture also allows definition of interfaces of this system with other systems. The composition represents a structured "parts list" - a hierarchy of components with their interfaces and interconnections. Edit the composition in the Composition Editor.



This example shows a motion control architecture, where a sensor obtains information from a motor, feeds that information to a controller, which in turn processes this information to send a control signal to the motor so that it moves in a certain way. You can start with this rough description and add component properties, interface definitions, and requirements as design progresses.

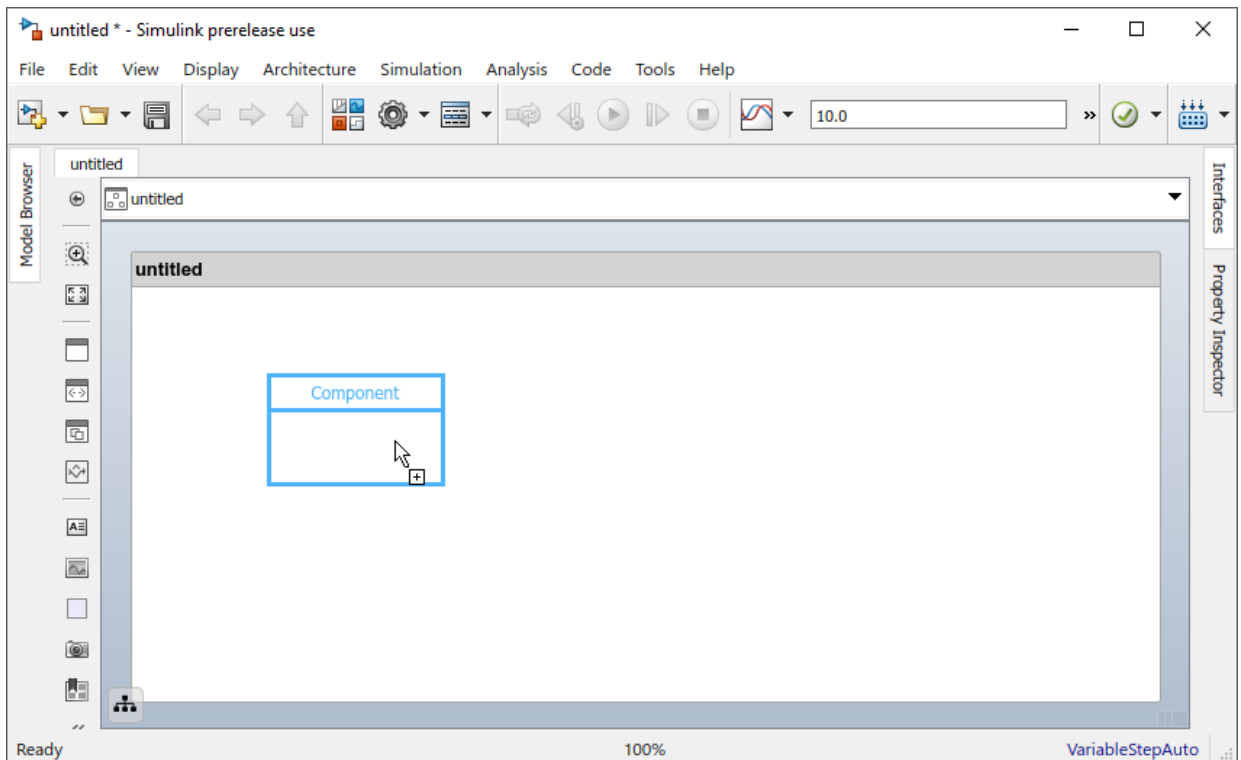
Components

A component is a non-trivial, nearly-independent, and replaceable part of a system that fulfills a clear function in the context of an architecture. The Component element in System Composer can represent a component at any level of the system hierarchy, whether it is a major system component that encompasses many subsystems, such as a controller with its hardware and software, or a component at the lowest level of hierarchy, such as a software module for messaging.

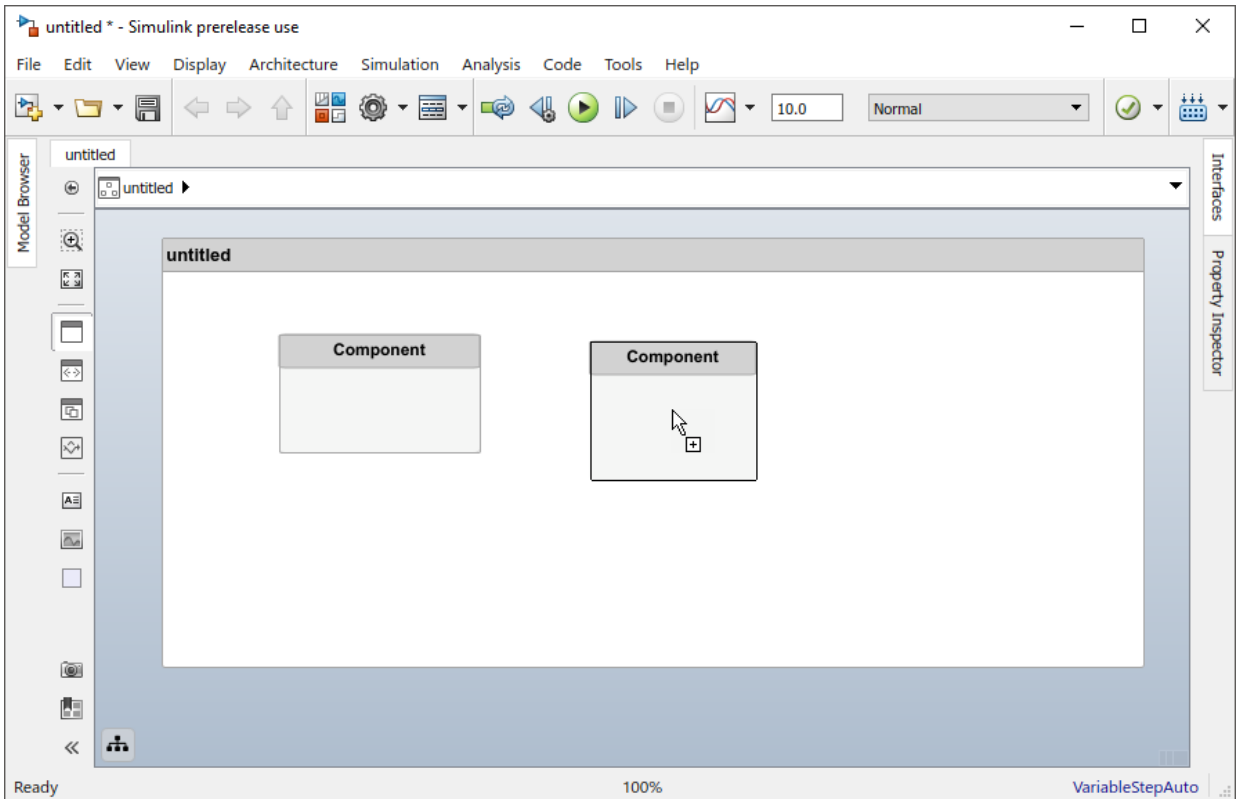
Add Components

Use one of these methods to add components to the architecture:

- Draw a component — Left-click on the canvas and hold down the mouse while dragging to create a rectangle. Release the mouse button to see the component outline. Click the light blue outline to commit.



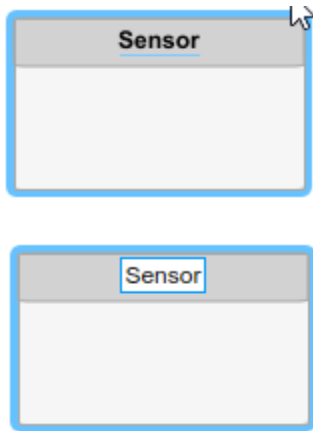
- Create a single component from the palette — Click the **Create Component** option in the palette and drag it onto the canvas.



- Create multiple components from the palette — double-click the **Create Component** icon and click the canvas at different locations. Right-click to stop creating components.

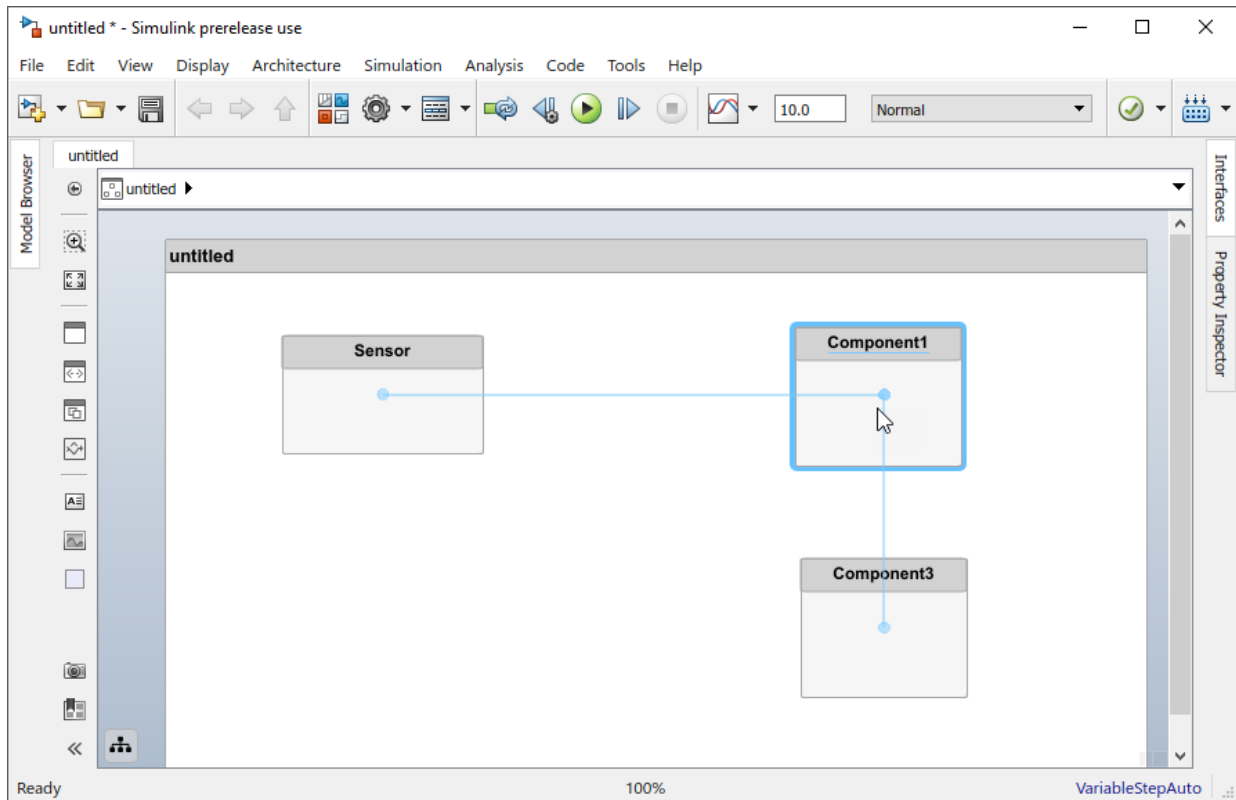
Name a Component

Each component must have a name that is unique within the same architecture level. The name of the component is highlighted upon creation so you can directly type the name. To change the name of a component later, click the component and then click its name.



Move a Component

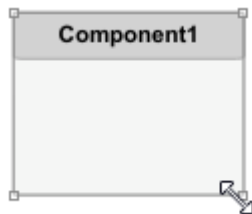
Move a component simply by clicking and dragging it. Blue guidelines may appear to help align the component with other components.



Resize a Component

Resize a component by dragging corners.

- 1 Hover the pointer over a corner to see the double arrow.



- 2 Left-click the corner and drag while holding the mouse button down. If you want to resize the component proportionally, hold the **Shift** button as well.



- 3 Release the mouse button when the component reaches the size you want.

Delete a Component

Click a component and hit **Delete** to delete it. To delete multiple components, select them while holding the **Shift** key down, then hit **Delete** or right-click and select **Delete** from the context menu.

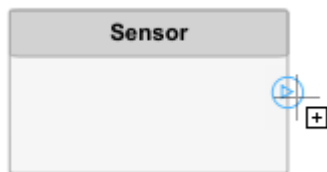
Ports

A port represents the connection point of a component to other components. For example, a sensor might have data ports to communicate with a motor and a controller. Its input port takes data from the motor, and the output port delivers data to the controller. You can specify data properties by defining an interface as described in “Define Interfaces” on page 3-2.

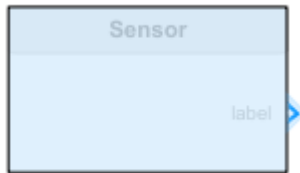
Add a Component Port

Represent the relationship between components by defining directional interface ports. You can organize the diagram by positioning ports on any edge of the component, in any position.

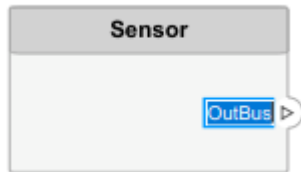
- 1 Hover at the left or right side of a component. A + sign and a port outline appear.





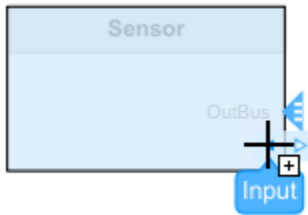
- Click the port outline. The component is shaded blue and a port shows with an arrow.



- Click the arrow to commit the port. You can also name the port at this point.



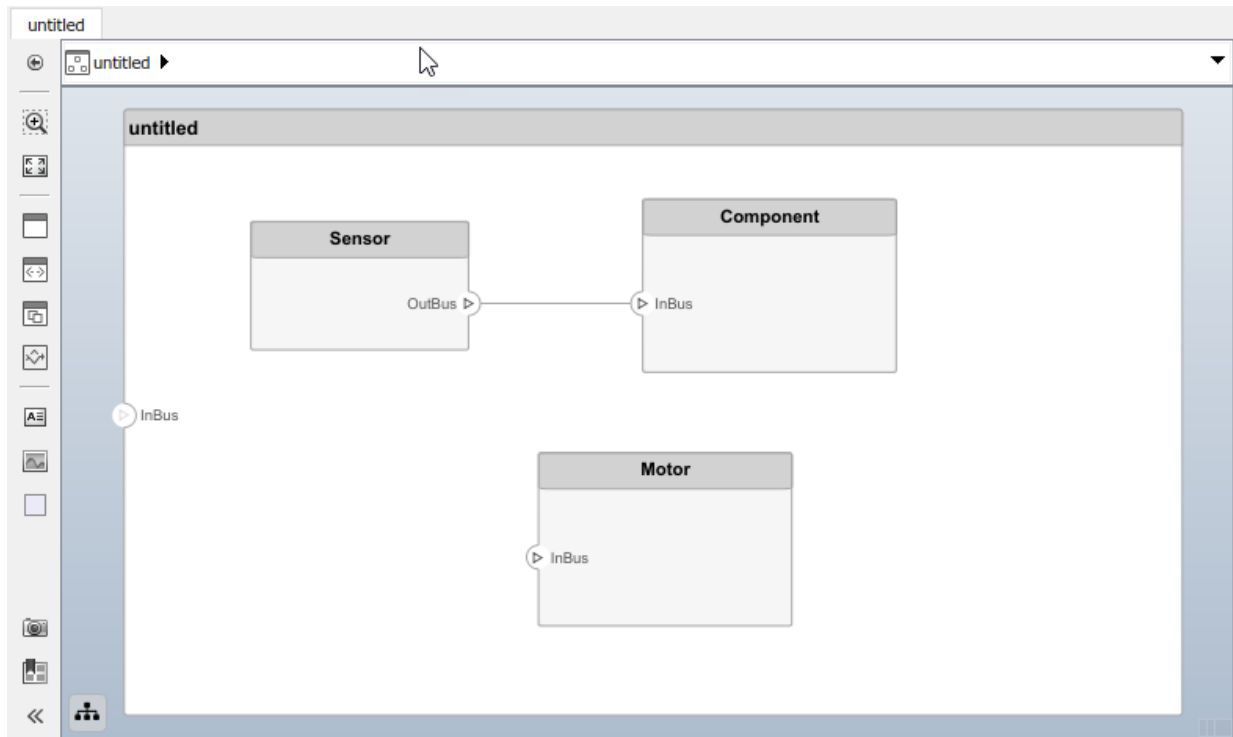
An output port is shown with the  and an input port is shown with the  icon. By default, a port created on the top or left edge of a component is an input port, and a port created on the bottom or right edge is an output port. To designate port direction at creation, hover over the arrow outline after you click the edge to see direction options. Select **Input** or **Output** before committing the port.



You can move any port to any component edge after creation.

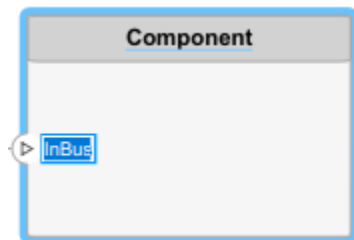
Add an Architecture Port

You can also create a port for the architecture that contains the components. These system ports carry the interface of the system with other systems. Hover on any edge of the system box and click when the + sign appears. Click the left side to create input ports and click the right side to create output ports.



Name a Port

Every port is created with a name. To change the name, click it and edit.



Ports of a component must have unique names.

Move a Port

You can move a port to any side of a component. Select the port and use arrow keys.

| Arrow Key | Original Port Edge | Port Movement |
|-----------|--------------------|--|
| Up | Left or right | If below other ports on the same edge, move up, if not, move to the top edge |
| | Top or bottom | No action |
| Right | Top or bottom | If to the left of other ports on the same edge, move right, if not, move to the right edge |
| | Left or right | No action |
| Down | Left or right | If above other ports on the same edge, move down, if not, move to the bottom edge |
| | Top or bottom | No action |
| Left | Top or bottom | If to the right of other ports on the same edge, move left, if not, move to the left edge |
| | Left or right | No action |

The spacing of the ports on one side is automatic. There can be a combination of input and output ports on the same edge.

Delete a Port

Delete a port by selecting it and clicking the **Delete** button.

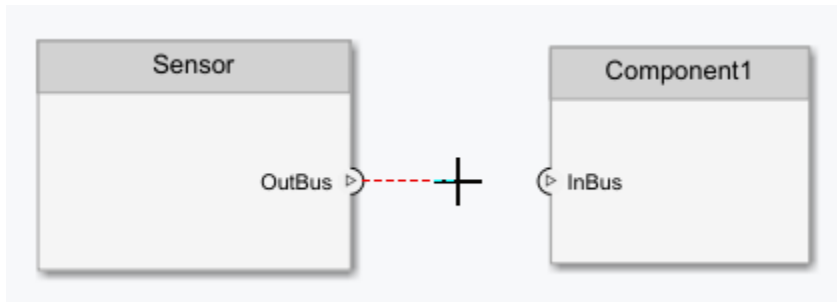
Connections

Connections are visual representations of data flow from an output port to an input port. For example, a connection from a motor to a sensor carries positional information.

Connect Existing Ports

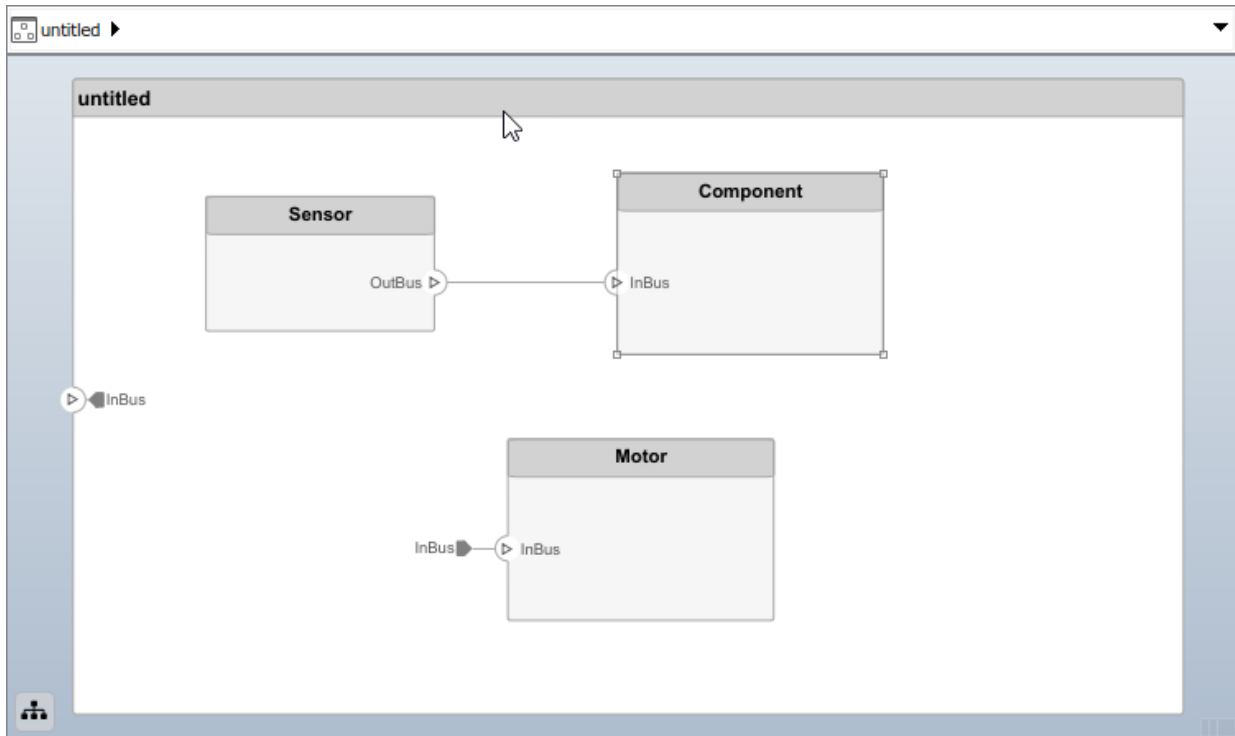
Connect two ports by dragging a line:

- 1 Click one of the ports.
- 2 Keep the mouse button down while dragging a line to the other port.
- 3 Release the mouse button at the destination port. A black line indicates the connection is complete. A red-dotted line appears if the connection is incomplete.



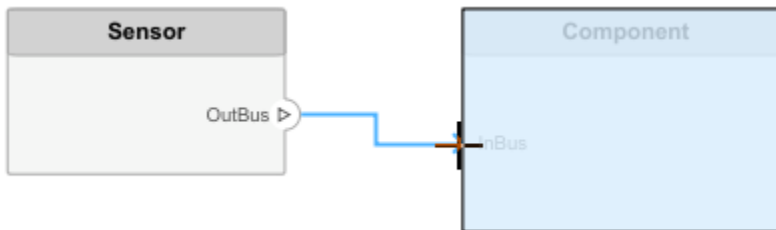
This procedure can be performed in both directions — input port to output port, or output port to input port. Nonetheless, you cannot connect ports that have the same direction.

A connection between an architecture port and a component port is shown with tags instead of lines.



Connect Components Without Ports

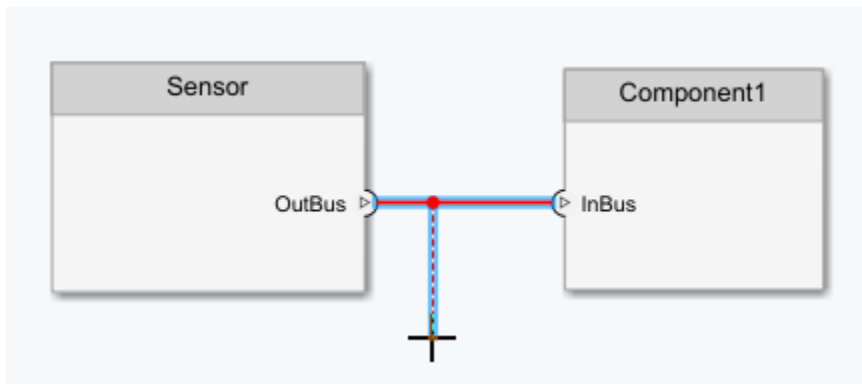
To quickly create ports and connections at the same time, drag a line from one component edge to another. The direction of this connection depends on which edges of the components are used - left and top edges are considered inputs, right and bottom edges are considered outputs. You can also perform this operation from an existing port to a component edge.



You can create a connection between an edge that is assumed to be an input only with an edge that is assumed to be an output. For example, you cannot connect a top edge, which is assumed to be an input, with another top edge, unless one of them already has an output port.

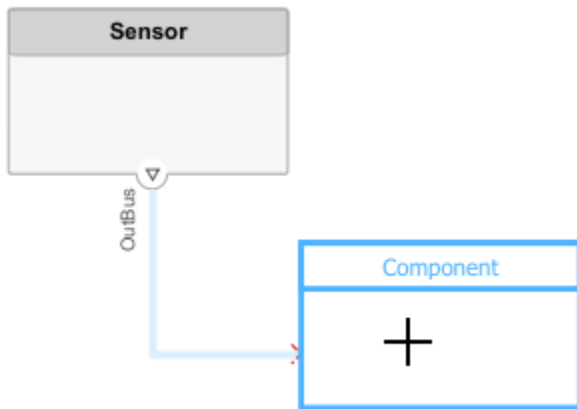
Branch Connections

Connect an output port to multiple input ports by branching a connection. To branch, right-click an existing connection and drag to an input port while holding the mouse button down. Release the button to commit the new connection.



Create New Components Through Connections

If you start a connection from an output port and release the mouse button without a destination port, a new component appears tentatively. Accept the new component by clicking it.



Importing Architectures

By combining the programmatic APIs of System Composer with MATLAB® support for loading and parsing many different file and databased formats, you can import external Architecture descriptions into System Composer. You can setup a profile with Stereotypes ahead of time to capture the Architecture properties represented in such descriptions. Subsequently, you can use MATLAB programming to create and customize the various Architectural elements through the set of programmatic APIs.

See Also

More About


- “Decompose and Reuse Components” on page 1-18
- “Link and Trace Requirements” on page 2-2
- “Define Interfaces” on page 3-2
- “Assign Interfaces to Ports” on page 3-7

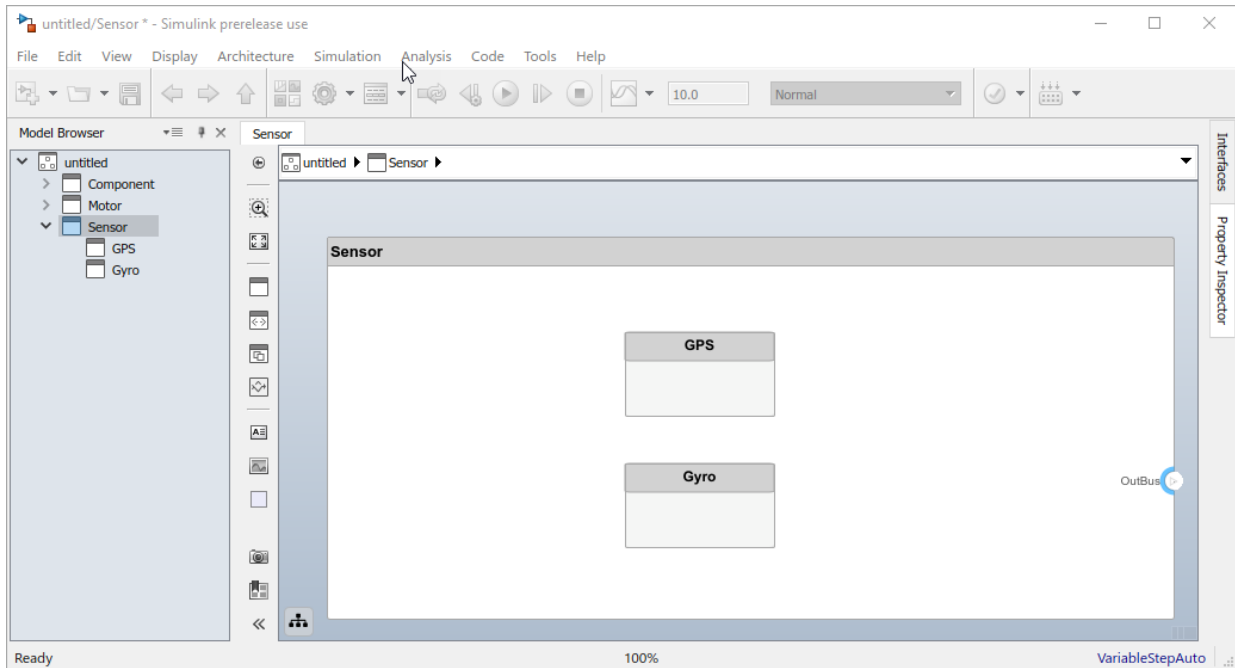
Decompose and Reuse Components

Every component in an architecture model can have its own design, or even several design alternatives. These designs can be architectures modeled in System Composer or behaviors modeled in Simulink®. Engineering systems often use the same component design in multiple places. A common component, such as power switch, can be part of all electrical components; You can reuse a component in System Composer within the same model as well as across architecture models.

Decompose a Component

A component can have its own architecture. Double-click a component to view or edit its architecture. The ports of a component appear as architecture ports when you view it at

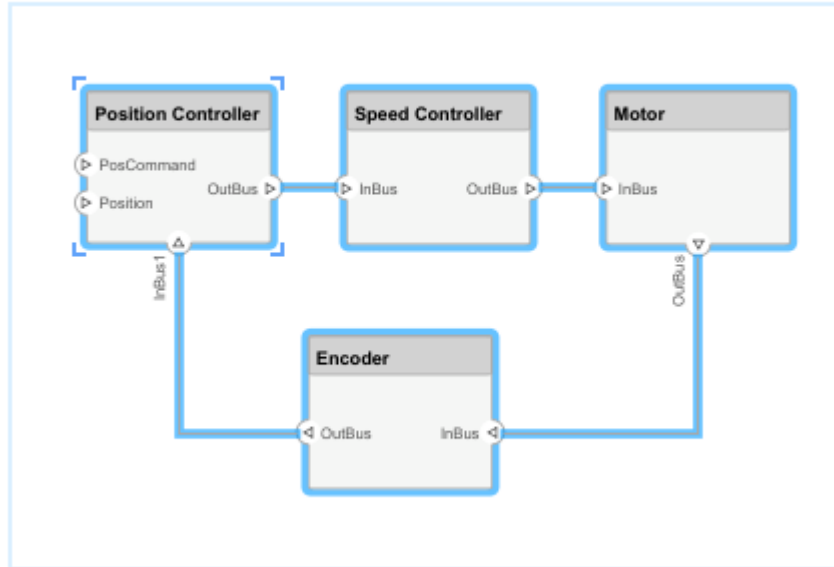
this level. You can use the navigation arrows  on the toolbar to move through the hierarchy. Use the Model Browser to view component hierarchy.



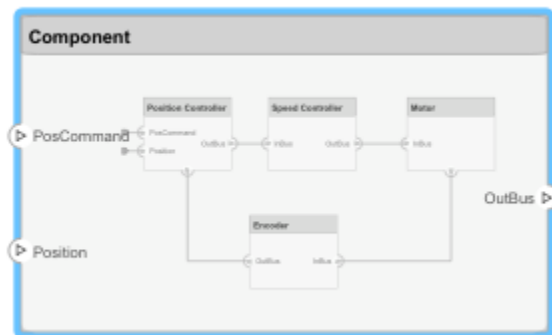
You can add components, ports, and connections at this level to define the architecture.

You can also make a new component from a group of components.

- 1 Select the components. Either click and drag a rectangle, or select multiple components by holding the **Shift** button down.



- 2 Create a component from the selected elements using **Architecture > Create Component**



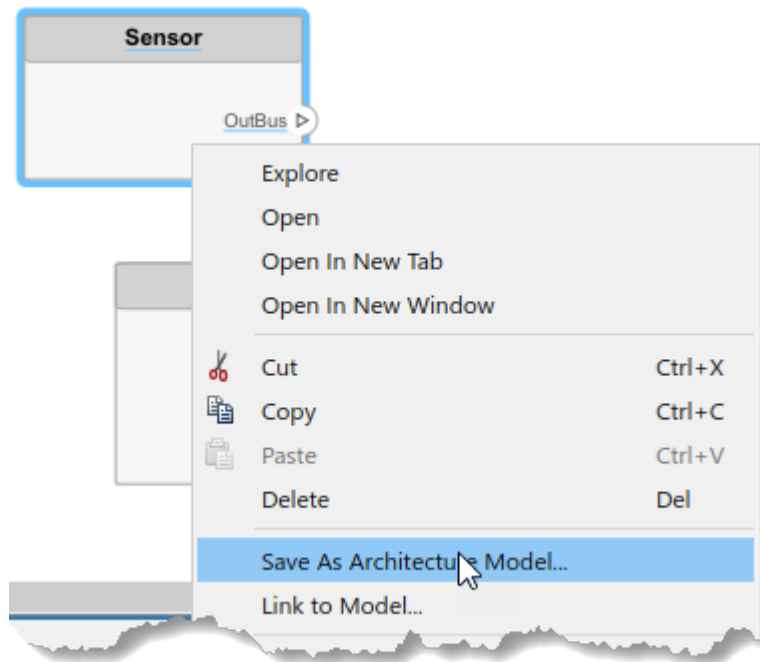
As a result, the new component has the selected components, their ports, and connections as part of its architecture. Any unconnected ports and connections to components outside of the selection become ports on the new component.

Any component that has its own architecture displays a preview of its contents.

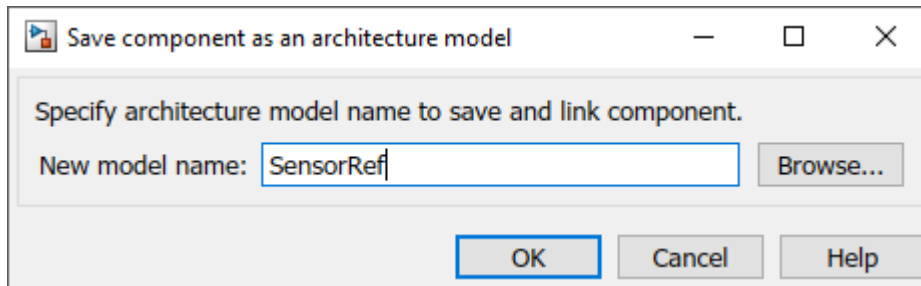
Create a Reference Architecture

Some projects use the same, detailed component in multiple places, and require the design of such a component to be tightly managed. You can create a reference architecture to reuse the architectural definition of a component in the same architecture model or across several architecture models. Create such a reference architecture using this procedure:

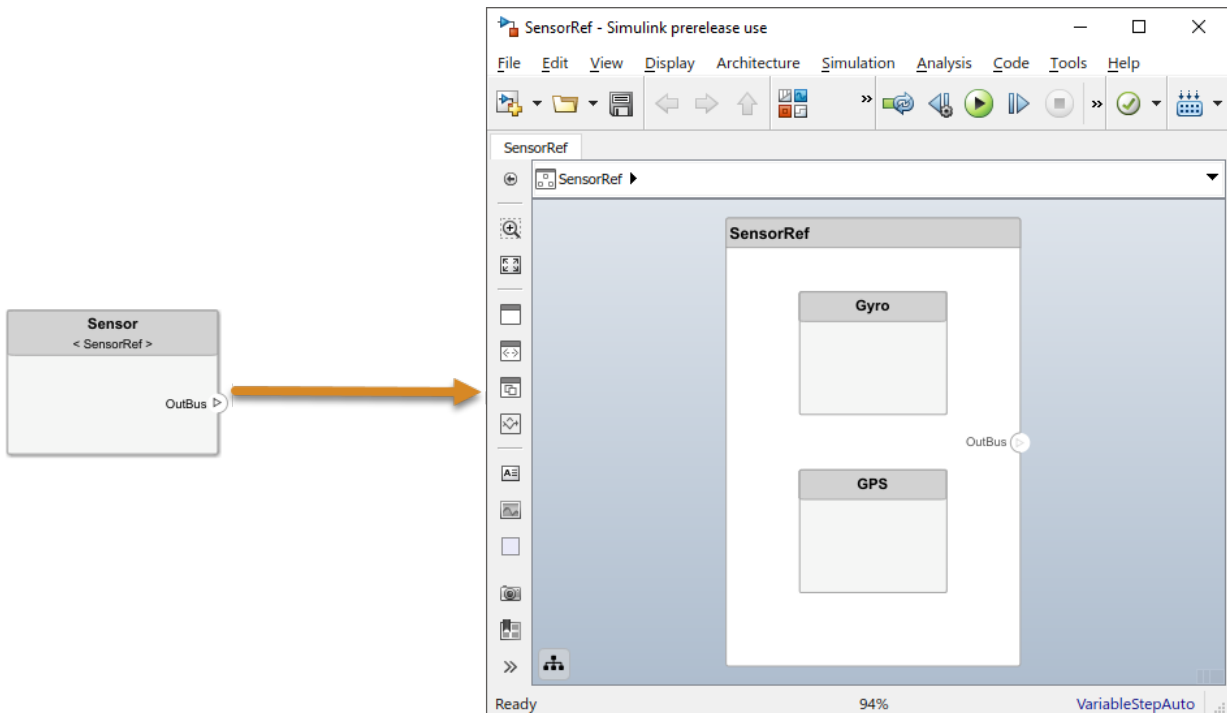
- 1 Right-click the component and select **Save as Architecture Model**.



- Provide a name for the model. By default, the reference architecture is saved in the same folder as the architecture model. Browse for or type the full path if you want to save it in a different folder.



System Composer creates an architecture model with the provided name, and links the component to the new model. The linked model is indicated in the name of the component between <> signs.

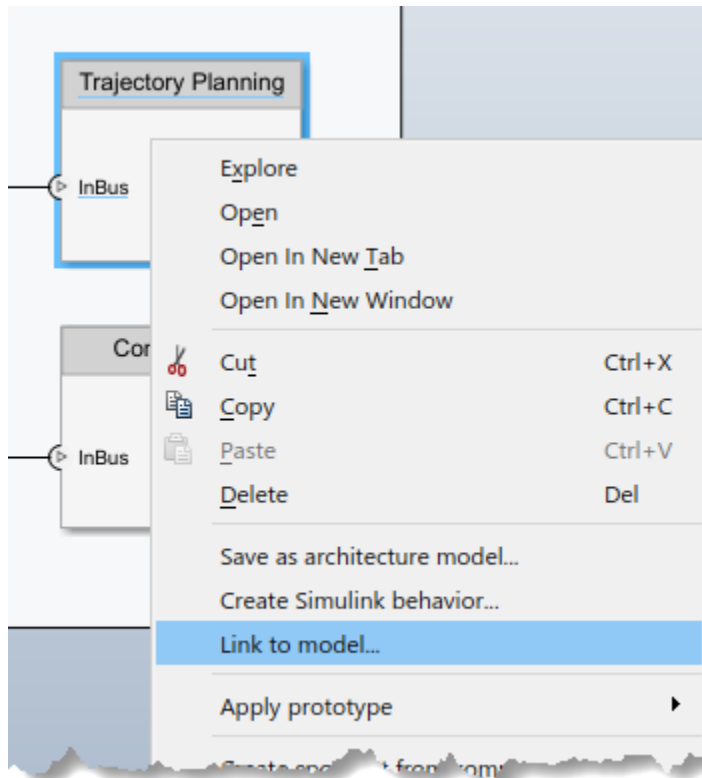


All architecture models can reference this new architecture model through linked components.

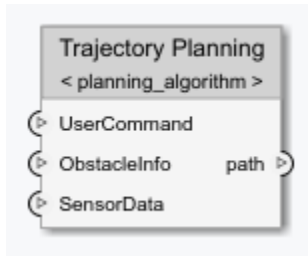
Use a Reference Architecture

You can use a reference architecture, saved in a separate file, by linking to it from a component. Right-click the component and select **Link to Model**. You can also use the **Create Reference** option in the element palette directly to create a component that uses a reference architecture.

To link a selected component to an existing architecture model, use **Architecture > Component > Link to Model** or right-click and select **Link to Model**.



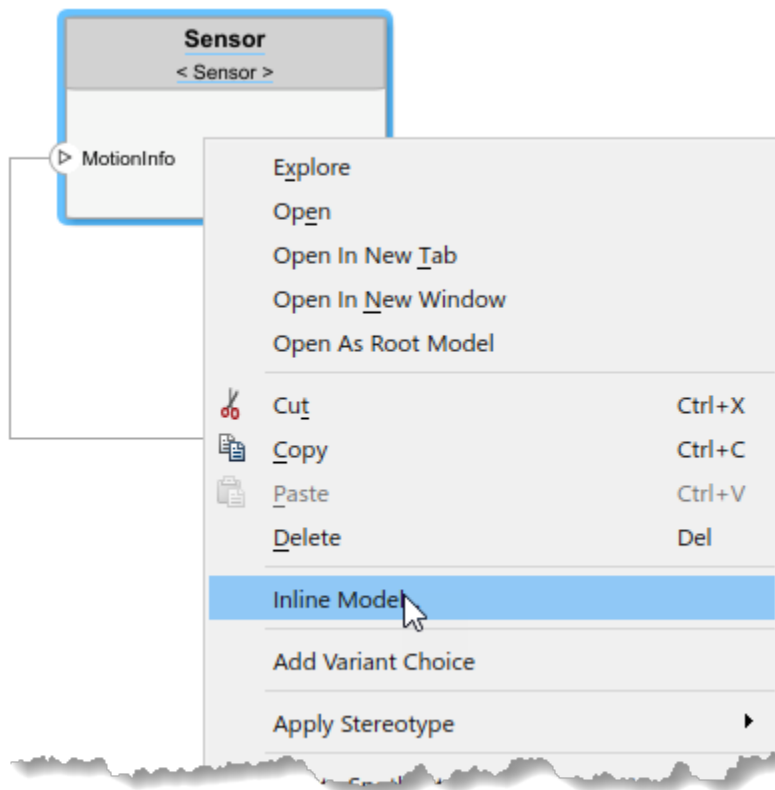
Provide the full path to the reference architecture. If the linked component has its own ports and components, this content gets deleted during linking and are replaced by that of the reference architecture. The ports of the linked component become the architecture ports in the reference architecture.



Any change made in a reference architecture is immediately reflected in the models that link to it. If you move or rename the reference architecture the link becomes invalid and the linked component shows an error. Link the component to a valid reference architecture to fix this.

Inline a Reference Architecture

Sometimes it is necessary to deviate from a reference architecture for a single component. For example, a comprehensive sensor model, referenced from a local component, may include too many features for the motion control architecture at hand and require simplification for that architecture only. In this case, you can inline the reference architecture to make local changes possible. Right-click a linked component and select **Inline Model**.



This operation provides two options:

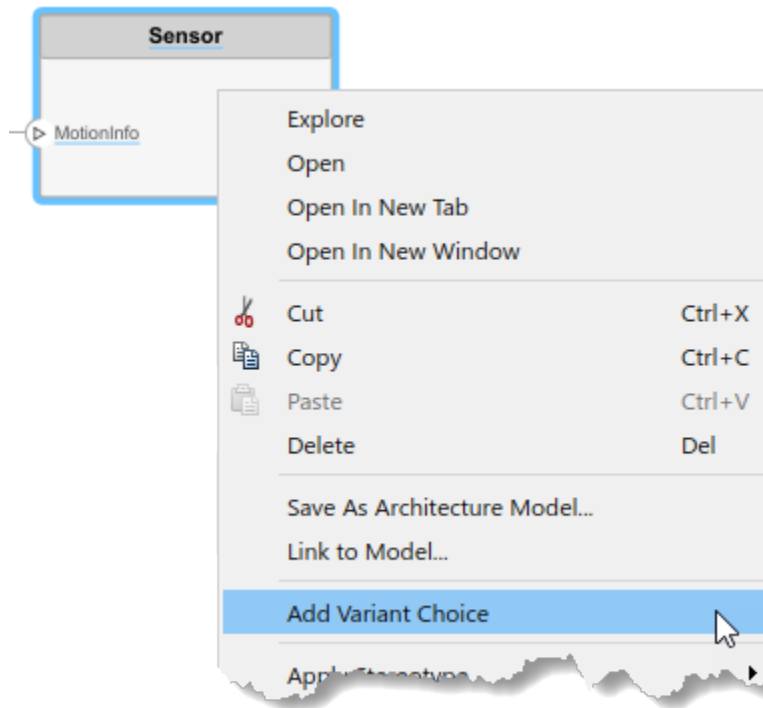
- Inline only interfaces: The ports and designated interfaces of the reference architecture are reflected on the component, but the composition is blank.
- Inline both interfaces and contents: Ports, interfaces, and subcomponents of the reference architecture are copied to the component.


Once the reference architecture is inlined, you can start making changes without affecting other architectures. However, you cannot propagate local changes to the reference architecture. If you link to the reference architecture again, local changes are lost.

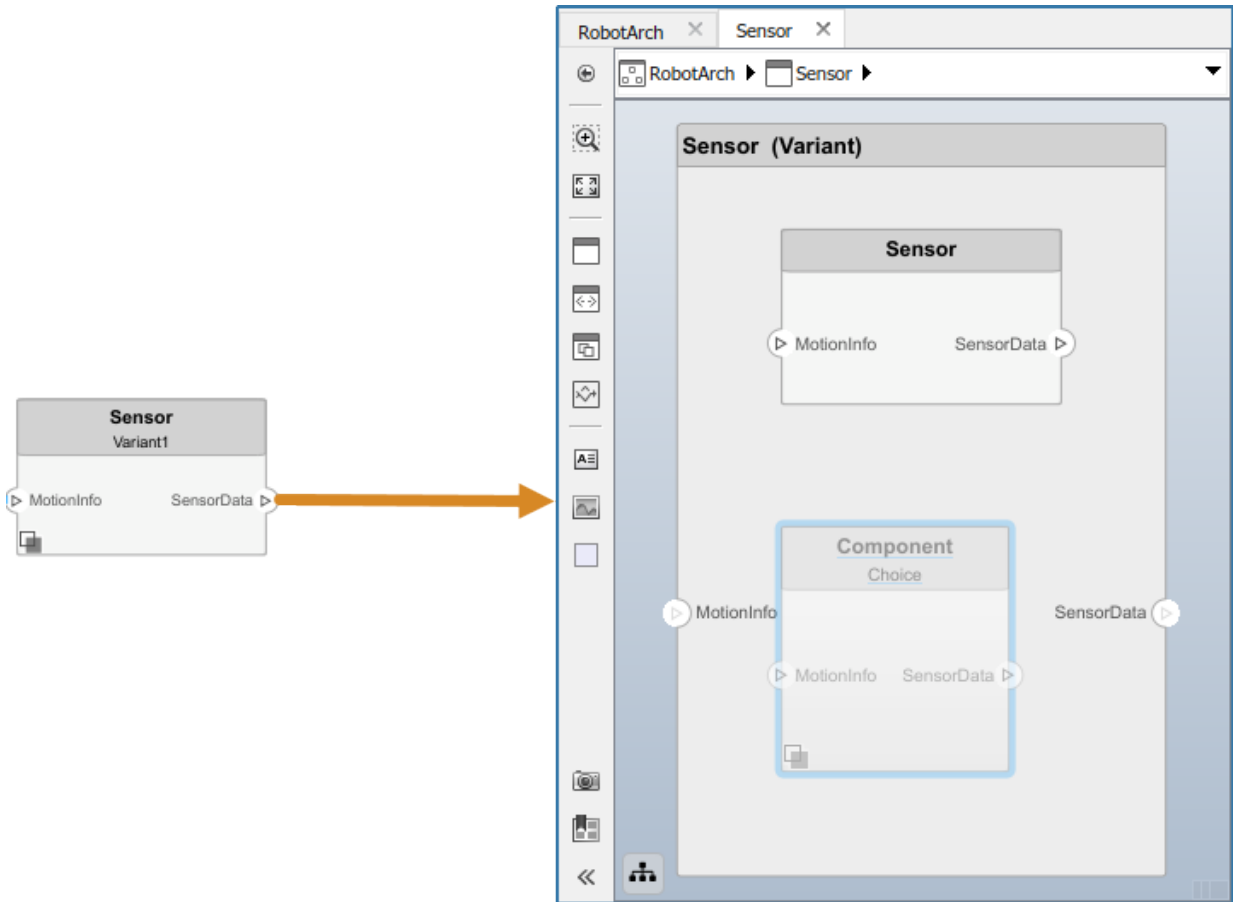
Create Variants

A component can have multiple design alternatives, or variants. You can model variations for any component in a single architecture model. You can define a mix of behaviors (defined in a Simulink model) and architectures (defined in a System Composer architecture model) as variant choices. For example, a component may have two variant options that represent two alternate structural decompositions.

Add variation to a component from the context menu or the **Architecture > Component** menu with the **Add Variant Choice** option.




The  badge on the component indicates that it is a variant and a variant choice is added to the existing composition. Double-click the component to see variant choices.

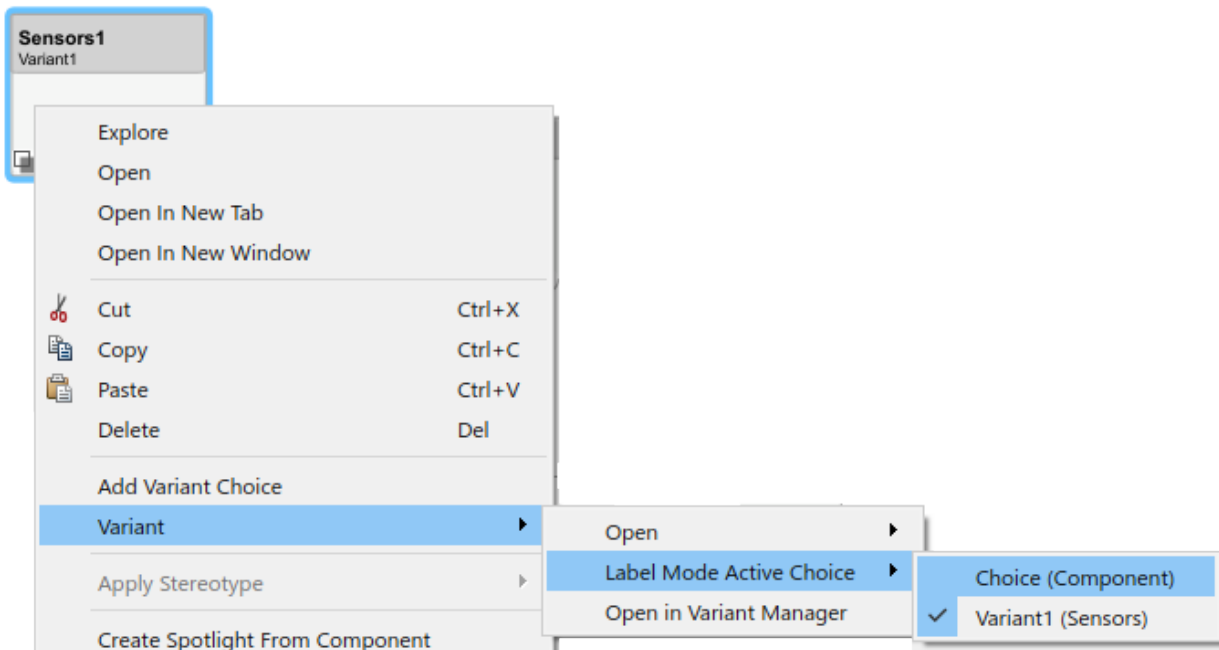


You can add more variant choices to a variant component using the **Add Variant Choice** option.

Open and edit the variant by selecting **Variant > Open > <variant_name>** from the component context menu.

You can also designate a component as a variant upon creation using the  object in the element palette. This creates two variant choices by default.

Activate a specific variant choice using the context menu of the block. Select The active choice is displayed in the header of the block.



See Also

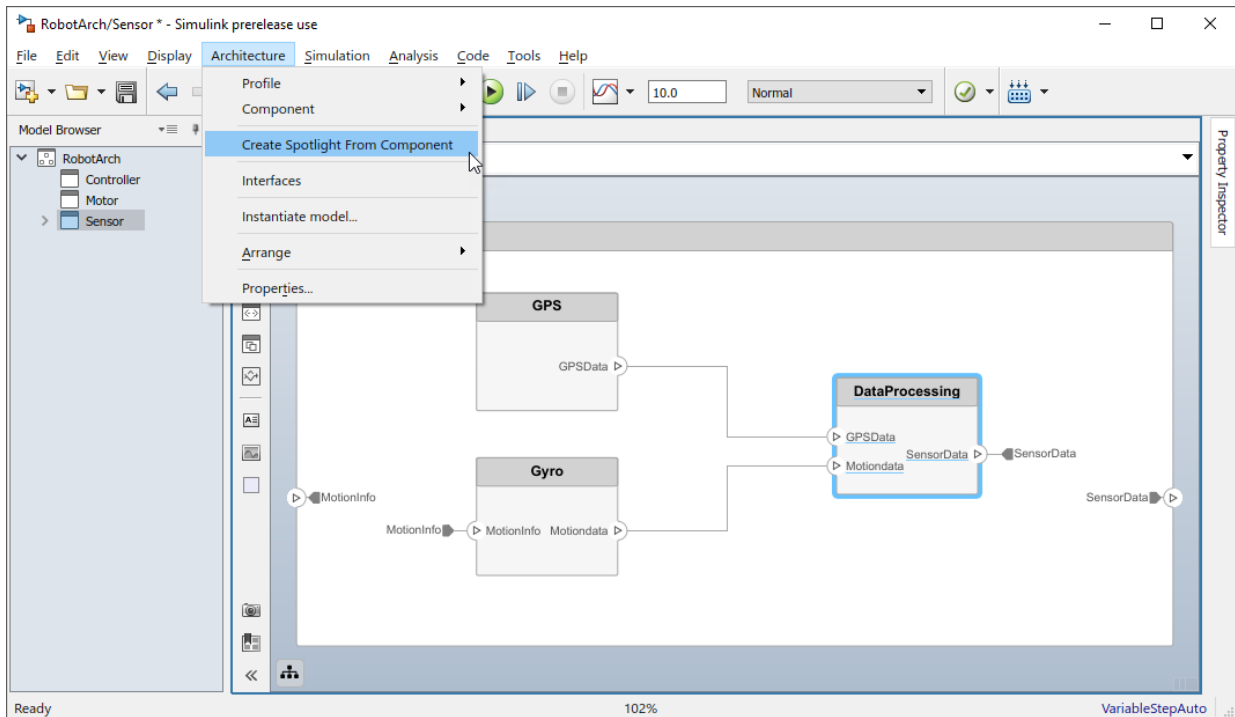
More About

- "Create a Simulink Behavior Model" on page 5-2
- "Link to an Existing Simulink Behavior Model" on page 5-4
- "Create Spotlight Views" on page 1-28

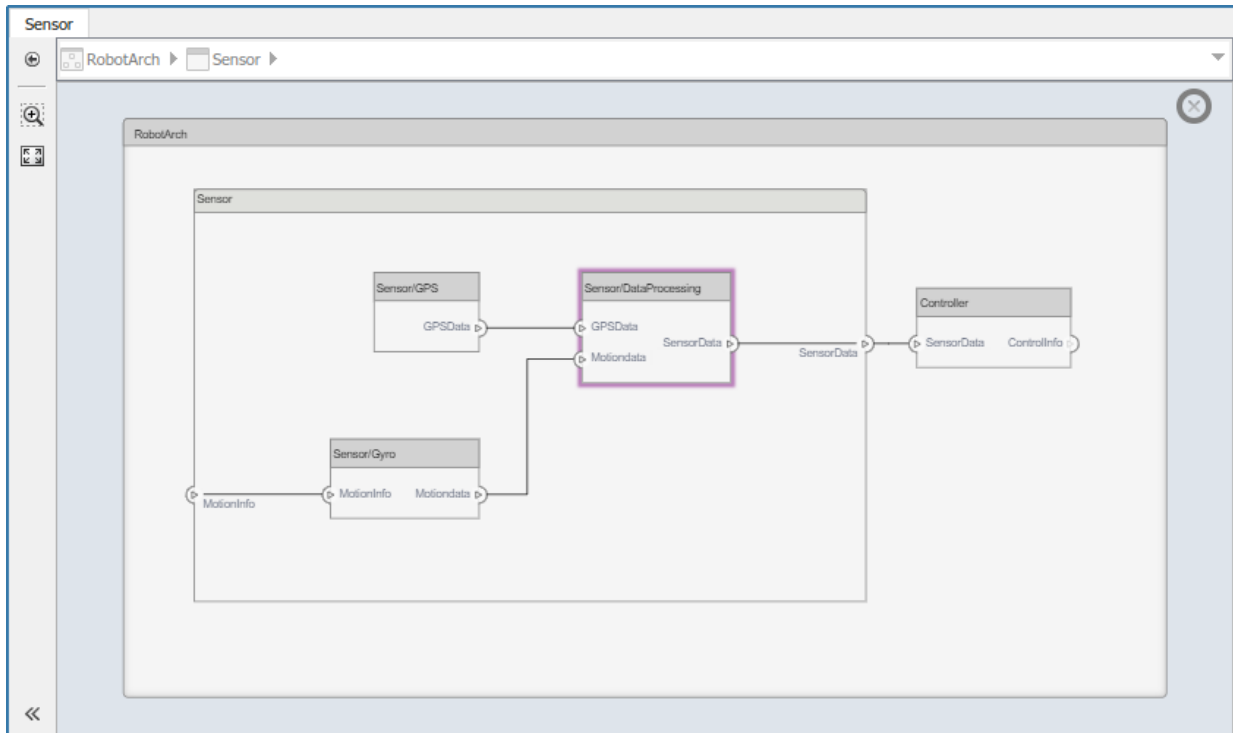
Create Spotlight Views


Any system being designed for a real application is usually very large and complex. It typically consists of many complex functions working together to fulfill the system requirements. In the process of designing and analyzing such architectures, you must understand existing components and what needs to be added. A Spotlight view is a simplified view of a model that captures the upstream and downstream dependencies of a specific component of interest.

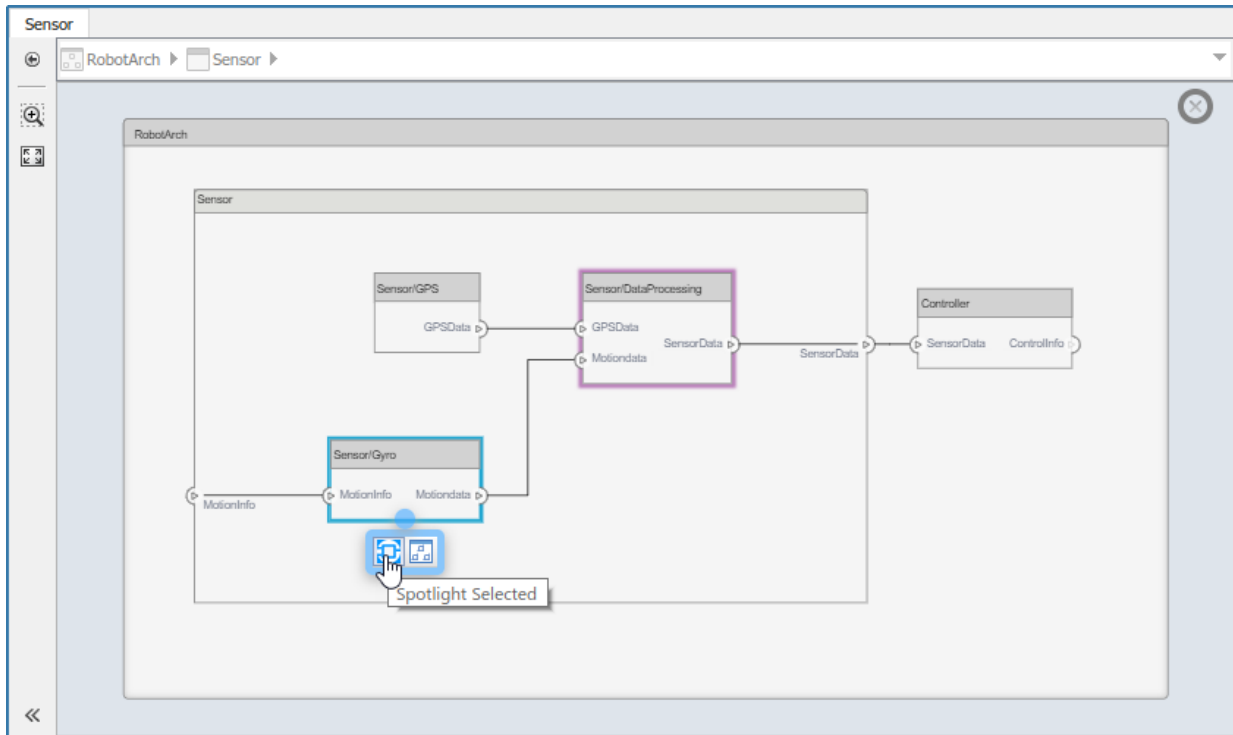
To create a spotlight from the composition, select the component of interest in the canvas and select **Create Spotlight from Component** either from the **Architecture** menu or the context menu.





The spotlight view launches and shows all model elements to which the component connects in a transparent hierarchy. The spotlight diagram is laid out automatically and cannot be edited.



While in the spotlight view, you can put another component in the spotlight. Select the component and click .



You can make the hierarchy and connectivity of a component visible at all times during model development by opening the spotlight view in a separate window. Show the spotlight view in a dedicated window by first selecting **Open in New Window** in the component context menu and then creating the Spotlight view. Spotlight views are dynamic. Any change in the composition refreshes any open spotlight views. Spotlight views are transient, they are not saved with the model.

You can return to the architecture model view by clicking the  icon. To view the architecture at the level of a particular component, select the component and click the  icon.

See Also

More About

- “Compose Architecture Visually” on page 1-2
- “Decompose and Reuse Components” on page 1-18

Requirements

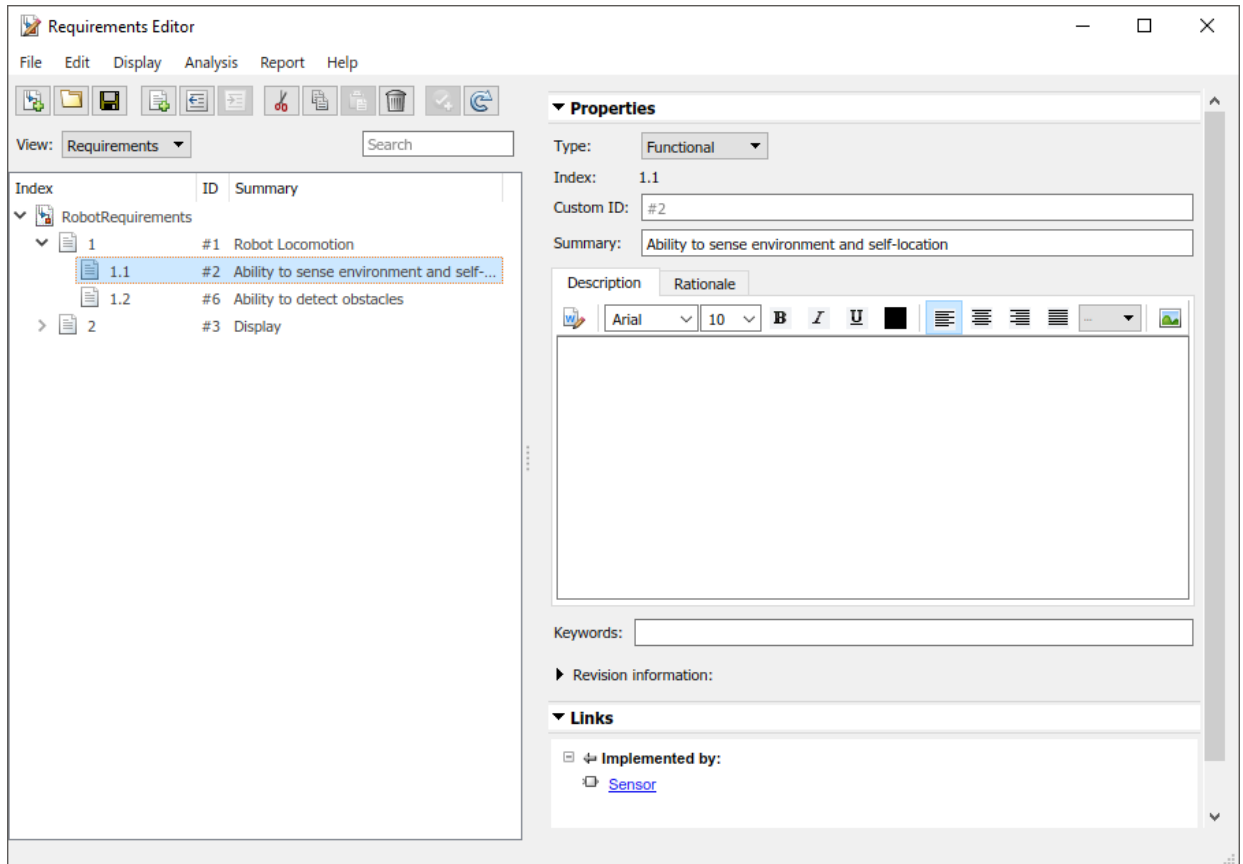
- “Link and Trace Requirements” on page 2-2
- “Manage Requirements” on page 2-9

Link and Trace Requirements

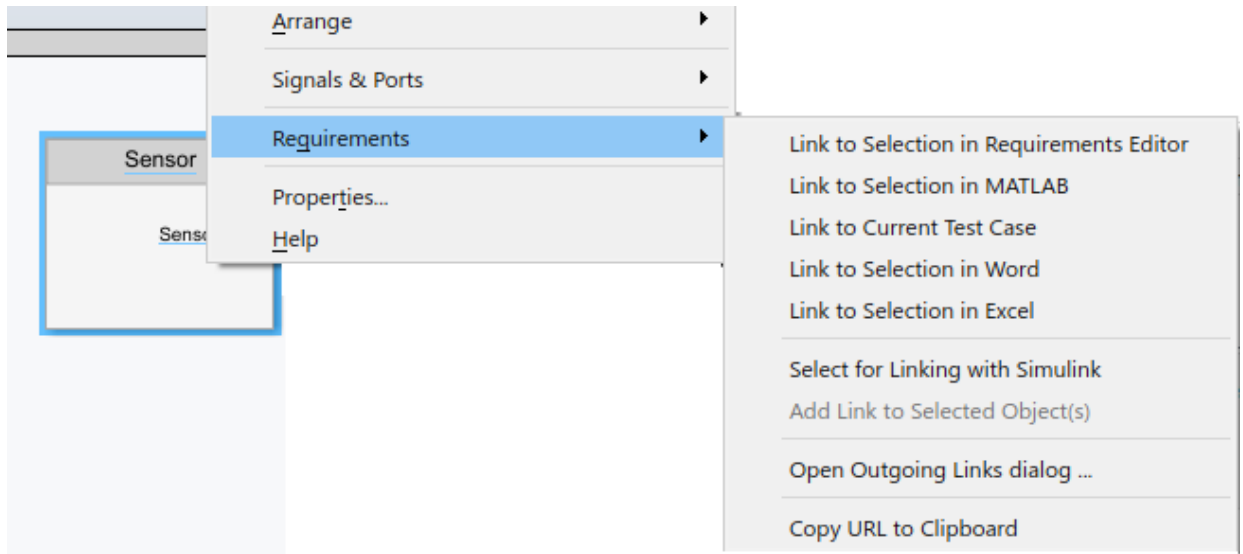
Allocate functional requirements to components to establish traceability. By creating a link between a component and the related requirement, you can track whether all requirements are represented in the architecture. You can also keep requirements and design in sync — for example, if a requirement changes or if the design warrants a revision of the requirements. You can link components to requirements in Simulink Requirements™, test cases in Simulink Test™, or selections in MATLAB, Microsoft® Excel®, or Microsoft Word.

Follow this procedure to link a component to Simulink Requirements:

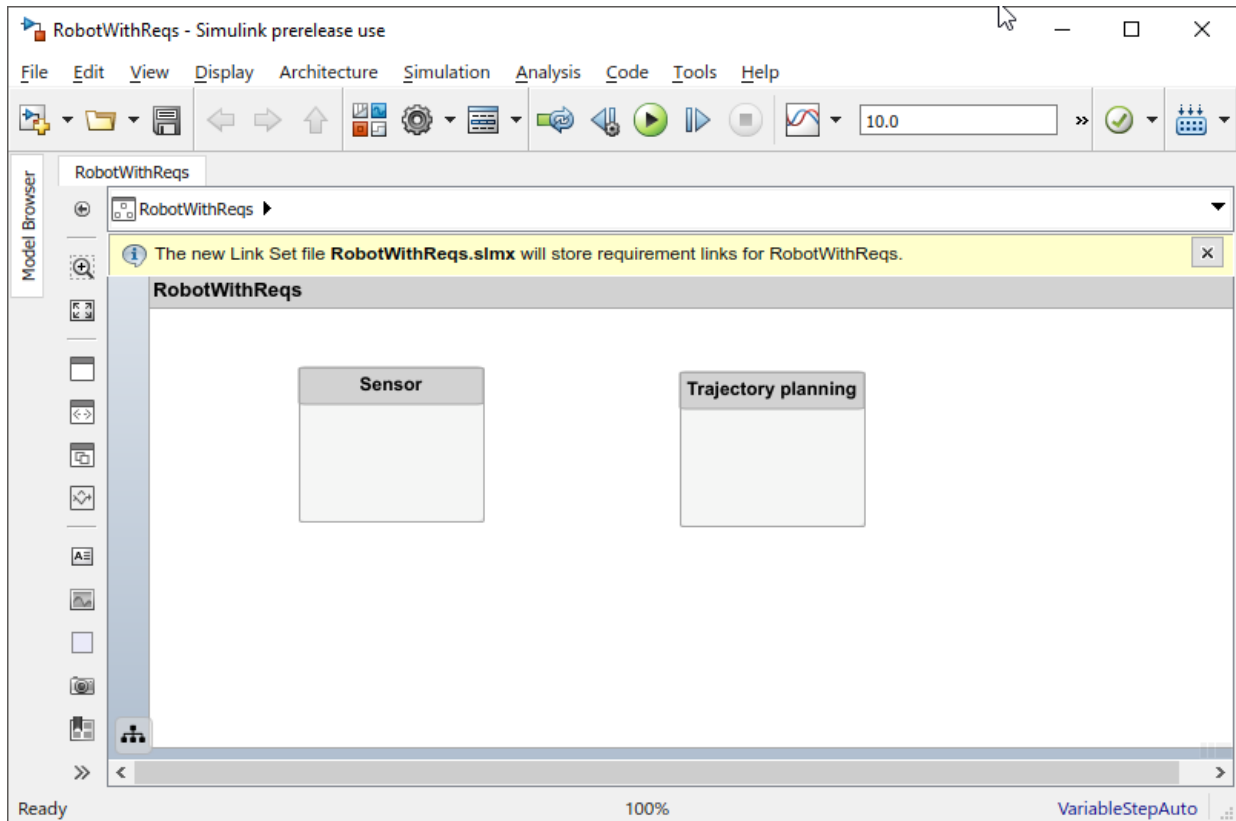
- 1 Save the architecture model.
- 2 Open the requirements in the Requirements Editor. The requirements file must be on the MATLAB path. Select the requirement to be linked:



- 3 Select the component to be linked in the architecture model. Right-click and select **Requirements > Link to Selection in Requirements Editor**.

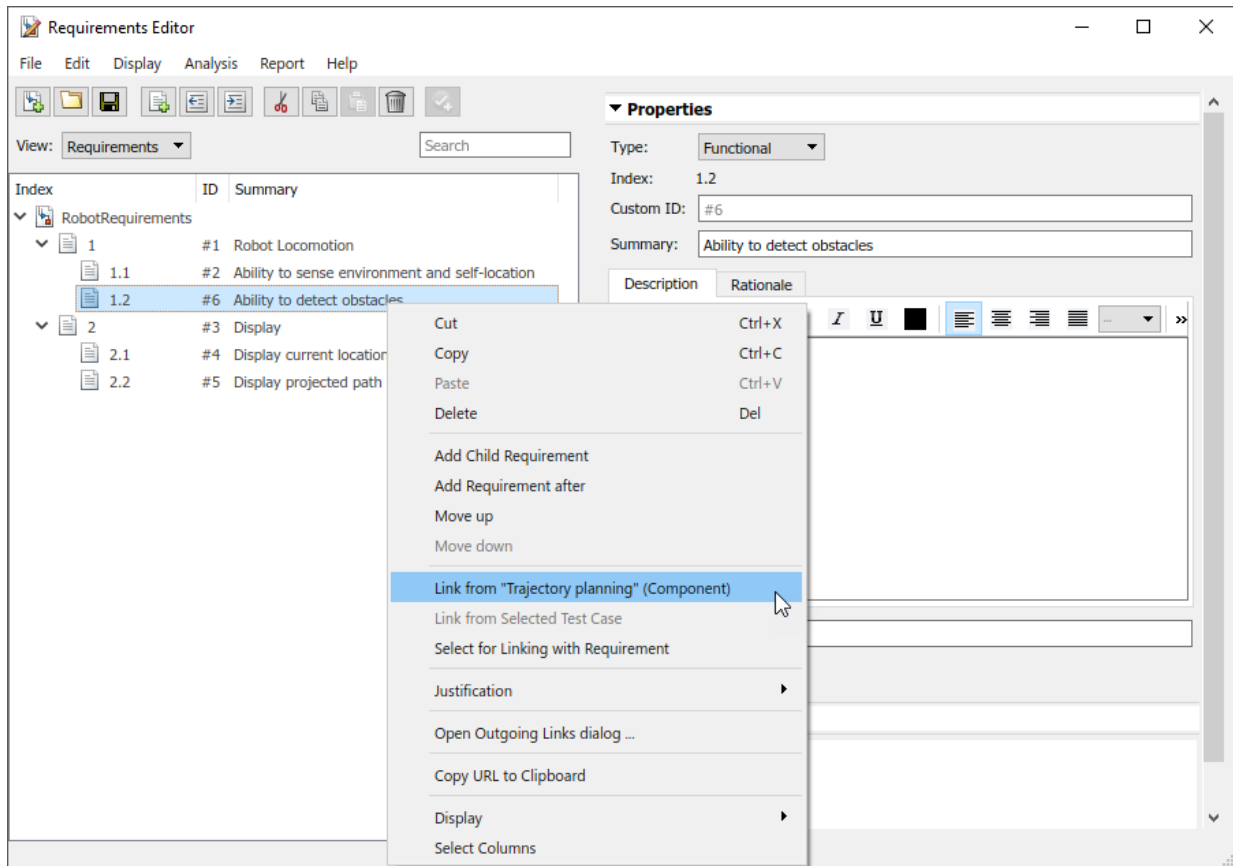


When you first link a requirement in an architecture model, a link set file with extension `.slmx` is created to store requirement links. The **Requirements** context menu displays the linked requirements.




You can also create a link using the Requirements Editor. First, select the component in the architecture model. Then, in the Requirements Editor, right-click the requirement and select **Link from <component_name> (Component)**.

2 Requirements

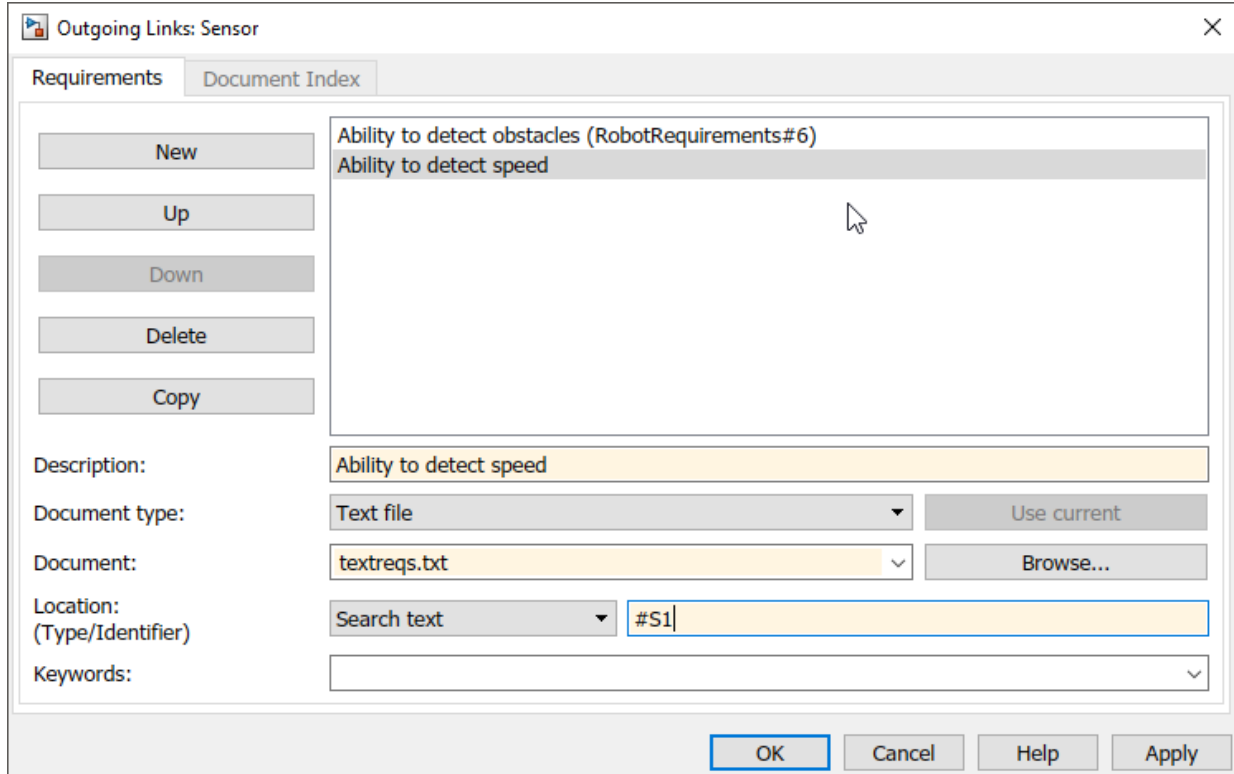


You can also create requirements links with blocks and subsystems in Simulink models. for more information, see “Link Blocks and Requirements” (Simulink Requirements).

A  badge on a component indicates that it is linked to a requirement. This badge also shows at the lower-left corner of the architecture model.



To trace requirement links to a component, select **Requirements > Open Outgoing Links dialog**. Here you can create new requirements, delete existing ones, and change their order.



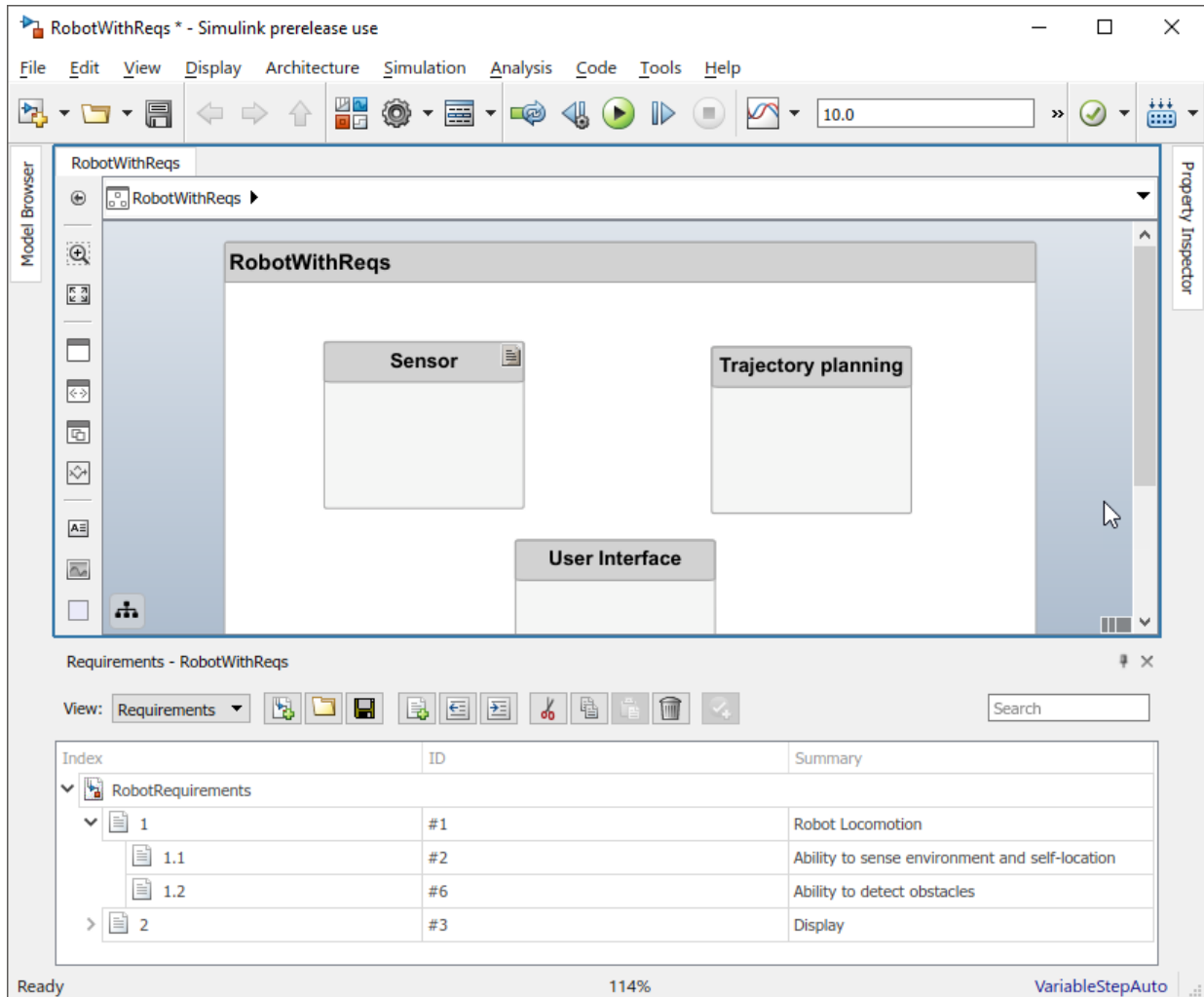
See Also

More About

- “Manage Requirements” on page 2-9
- “View Linked Requirements in Models and Blocks” (Simulink)

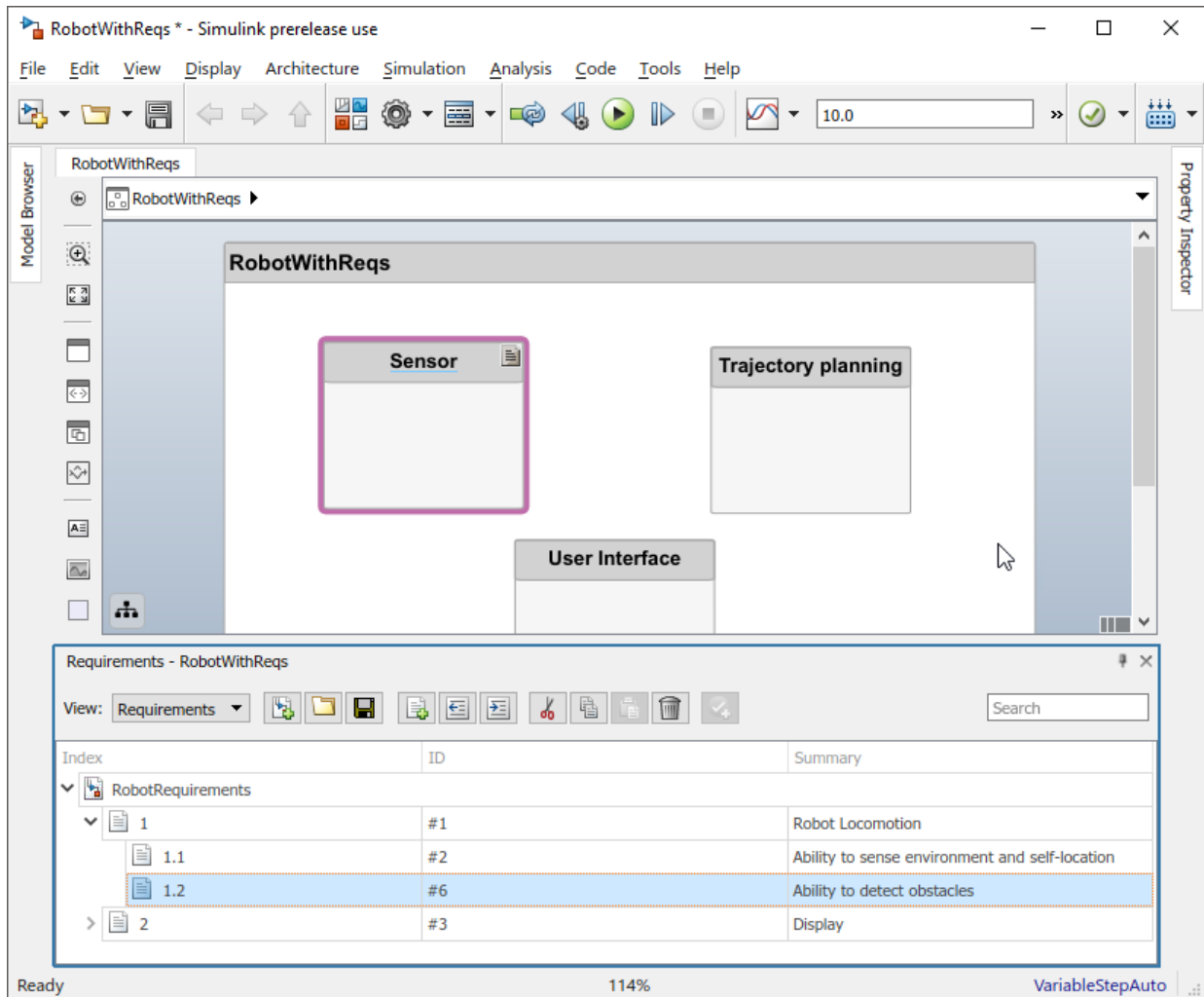
Manage Requirements

Manage requirements and architecture model together in the Requirements Perspective. Select **Analysis > Requirements > Requirements Perspective**.



When you click a component in the Requirements Perspective, linked requirements are highlighted. Conversely, when you click a requirement, the linked components are shown.

2 Requirements



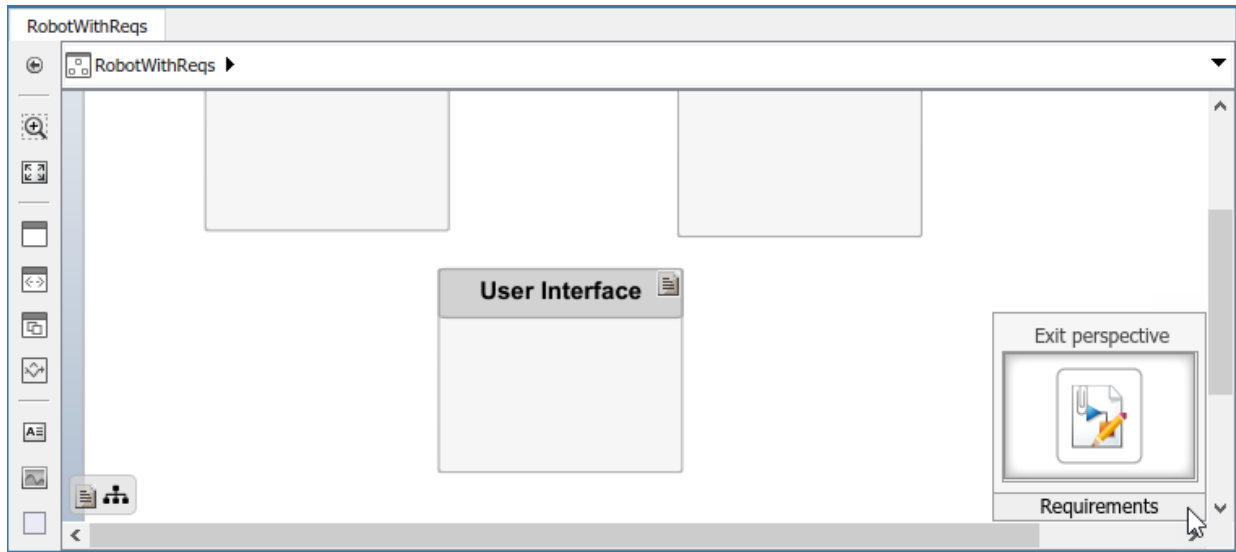
You can drag a requirement onto a component to directly create a link.

The screenshot shows the Simulink Requirements tool interface. The main window displays an architecture model with a 'User Interface' block and a requirement '#3: Display' block. An 'IMPLEMENTS' relationship is shown between them. A 'Requirements - RobotWithReqs' window is open at the bottom, displaying a table of requirements.

| Index | ID | Summary |
|-------------------|----|--|
| RobotRequirements | | |
| 1 | #1 | Robot Locomotion |
| 1.1 | #2 | Ability to sense environment and self-location |
| 1.2 | #6 | Ability to detect obstacles |
| 2 | #3 | Display |

You can close the annotation that shows the link as necessary, this does not delete the link.

You can exit the Requirements perspective by clicking the perspectives menu on the lower-right corner of the architecture model and selecting **Exit perspective**.



For more information on managing requirements, see “Manage Navigation Backlinks in External Requirements Documents” (Simulink Requirements).

See Also

More About

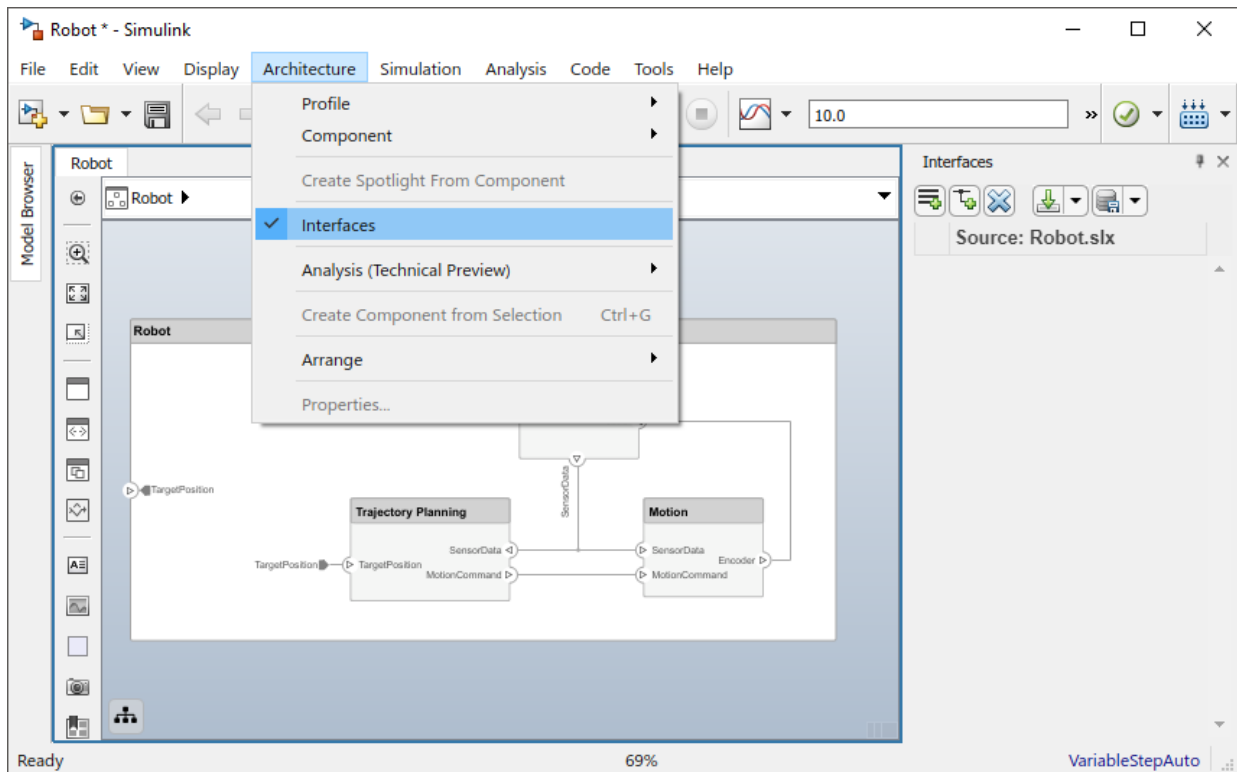
- “Link and Trace Requirements” on page 2-2
- “Link Blocks and Requirements” (Simulink Requirements)

Interface Management

- “Define Interfaces” on page 3-2
- “Assign Interfaces to Ports” on page 3-7
- “Save and Link Interfaces” on page 3-11

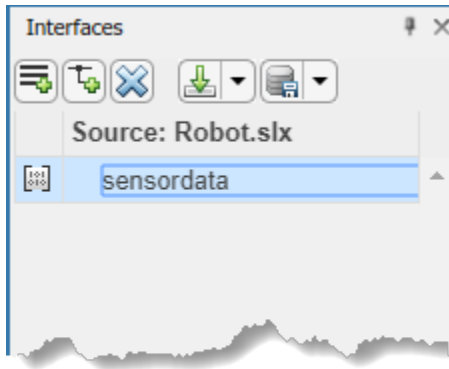
Define Interfaces


A system engineering solution includes formal definition of the interfaces between components. A connection merely shows that the two components has an output-to-input relationship; an interface defines the type, dimensions, units, and structure of the data. You can describe interfaces using the Interface Editor. You can show or hide the Interface Editor using **Architecture > Interfaces**.

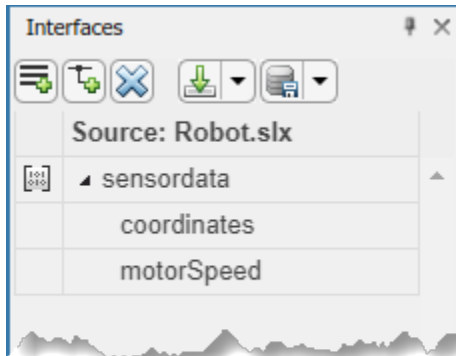



Create Interface

To add a new interface definition, click the  icon. Name the interface.

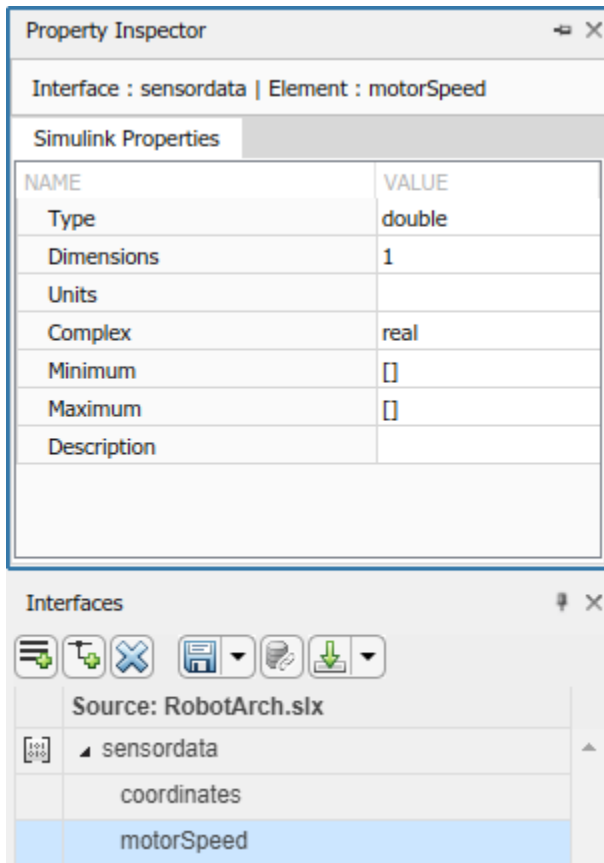


Click the  icon to add an element to the interface. Change the name. Interface and element names must be valid variable names.



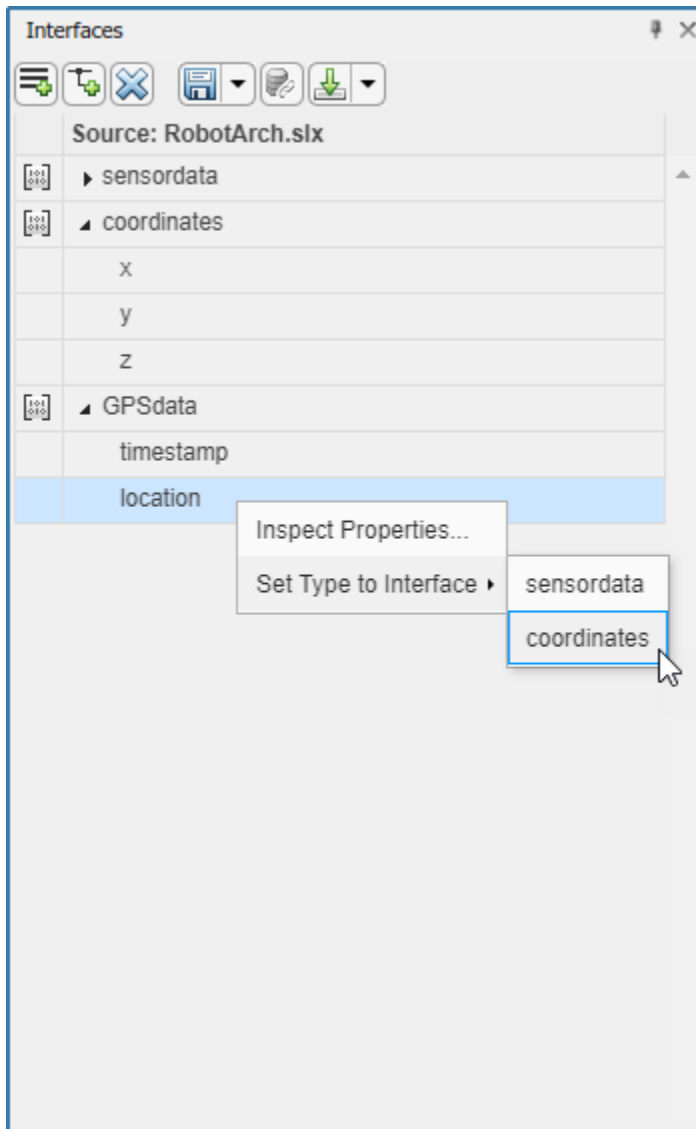
You can delete interfaces and elements in the Interface Editor using the  button.

You can view and edit the properties of an element in the Property Inspector. Open the Property Inspector using **View > Property Inspector** or right-click the interface element and select **Inspect Properties**.



A hierarchical interface is one that contains another interface. Create a hierarchical interface by assigning the type of an interface element to another interface.

For example, let `coordinates` be an interface that consists of `x,y,z` coordinates. GPS data includes location information and a timestamp. If the location data is in the same format as the `coordinates` interface, then you can set its type to `coordinates`. Right-click the location element and select **Set 'Type' > coordinates**. The available interface options include all interfaces in the model, except the parent of the element.



The defined interfaces become part of the model data dictionary.

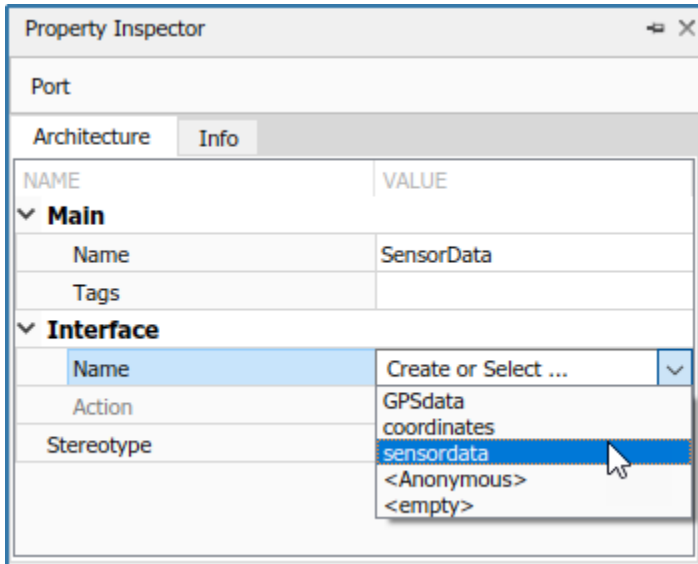
See Also

More About

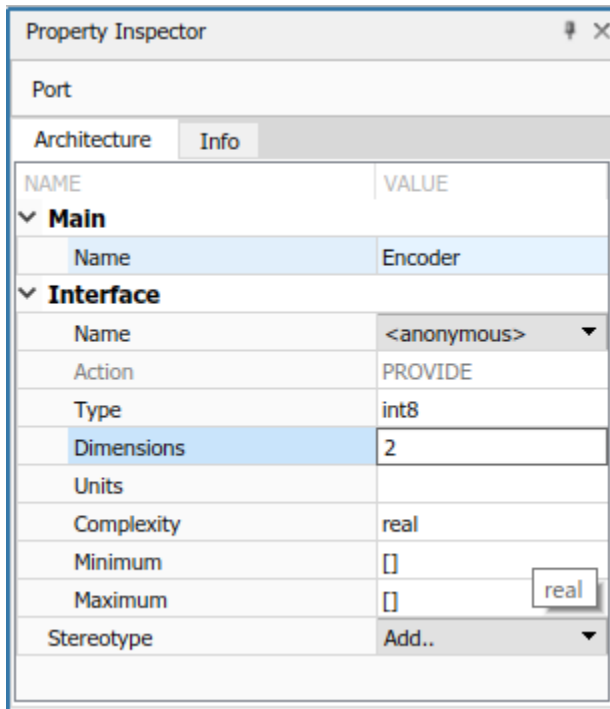
- “Assign Interfaces to Ports” on page 3-7
- “Save and Link Interfaces” on page 3-11

Assign Interfaces to Ports

Associate a port with an interface using Property Inspector. Highlight the port to show its properties. Expand **Interface**, and select the interface in the **Name** drop-down menu.

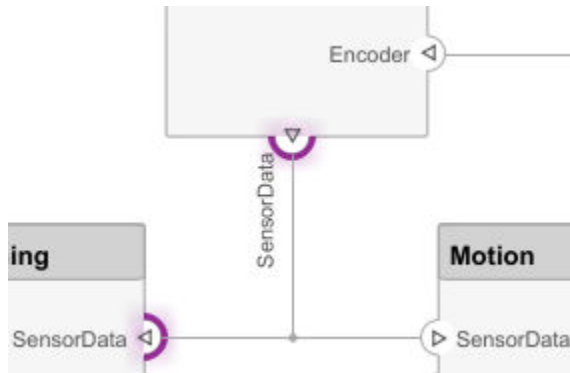


You can select an interface in the model data dictionary (see “Define Interfaces” on page 3-2), or create an anonymous interface - unstructured data whose properties are valid for that port only. An anonymous interface does not have structure, but has prescribed properties such as Type, Dimensions, etc. You can edit the properties of the anonymous interface in the Property Inspector.

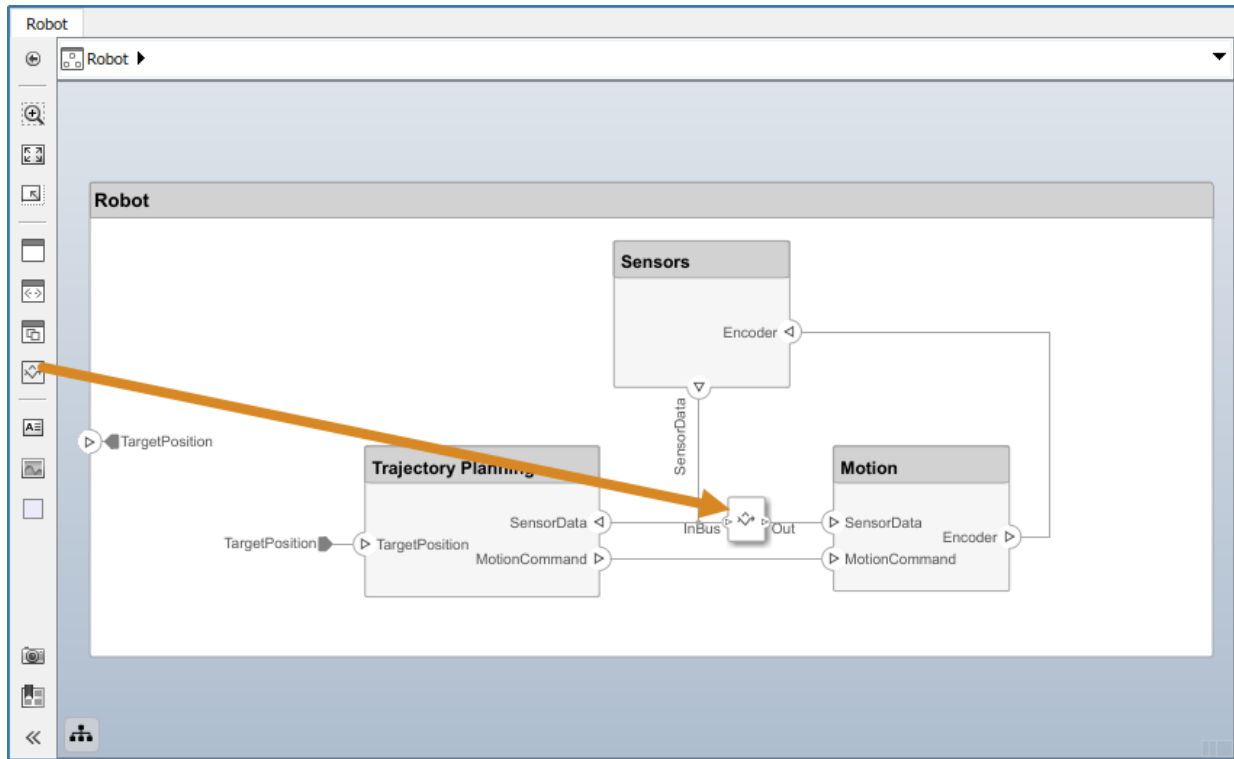


Multiple ports, whether they are connected or not, can use the same interface definition. When you assign an interface to a port, it is automatically propagated to the connected ports, provided they do not already have assignments.

Highlight the ports that use an interface definition by clicking the interface name in the Interface Editor.



It is possible to have different interfaces assigned to the source port and destination port of a connection. This could represent an intermediate point in design, where components from different sources are brought together. You can use an Interface Adapter block from the component palette to connect components with different interfaces.



Change the number of input and output ports of an interface adapter the same way you add and remove component ports. See “Ports” on page 1-10.


See Also

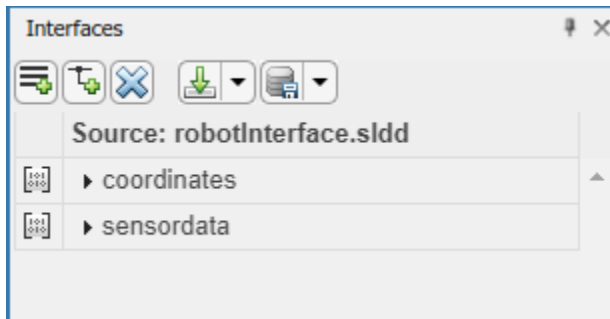
More About

- “Define Interfaces” on page 3-2


Save and Link Interfaces

Engineering systems often use the same interface definition for multiple projects. For example, a data feed can be standardized across a system to have the same structure to facilitate processing. By default, interfaces are stored within the architecture model and are not visible outside the model. However, if you want to create a group of models where components of one model refer to another model, you can take advantage of the data dictionary to store interfaces so that they may be shared across models.

Use the  menu to save an interface to a new or existing data dictionary. Create a new data dictionary by selecting **Save and link > New dictionary**. Provide a dictionary name.



You can also add the interface definitions in the model to an existing data dictionary by selecting **Save and link > Existing dictionary**.

Use the  button to import interface definitions from a Simulinkbus object, either from a MAT-file or the workspace.

More About

- “Define Interfaces” on page 3-2
- “Save and Import Bus Objects” (Simulink)

Define Architectural Properties

- “Define Profiles and Stereotypes” on page 4-2
- “Use Stereotypes and Profiles” on page 4-8

Define Profiles and Stereotypes

To verify structural and functional requirements, you must capture non-functional properties on elements in an architecture model. For example, if there is a limit on the total power consumption of a system, the model must capture the power rating of each electrical component. This requires extending built-in model element types with properties relevant to requirements, in this case, an electrical component type as an extension of components. You can introduce a self-consistent domain of model element types into System Composer using a set of stereotypes, called a profile.

System Composer provides the following architectural model elements to describe an architecture model:

- Component
- Port
- Connection

You can view the properties of each element in the architecture model using the Property Inspector. Open Property Inspector using **View > Property Inspector**.

You can define stereotypes to extend built-in elements and capture additional data about an element. Element stereotypes define the class of the elements to which they apply. For example, an `MechanicalComponent` stereotype with properties such as `Weight` and `Volume` applies only to components.

A stereotype does not have to define a class. For example, a `ProjectItem` stereotype can add generic properties such as catalog number or unit cost, a `BorrowedItem` stereotype can add properties such as `BorrowedSource` and `ReturnDeadline`. A model element can have multiple stereotypes.

Stereotypes can extend other stereotypes to include their properties. For example, a `UserInterface` stereotype can be an extension of a `SoftwareComponent` stereotype, and add a property called `ScreenResolution`.

You can collect stereotypes in profiles. You author profiles using the Profile Editor. Profiles are saved separately from the architecture model and are available to all architecture models.

When you create a profile, you define:

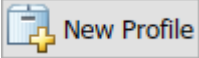
- Stereotypes — Customize built-in model element types
- Property sets — Add analysis properties to an architecture model element
- Data types, dimensions, etc — Define property values

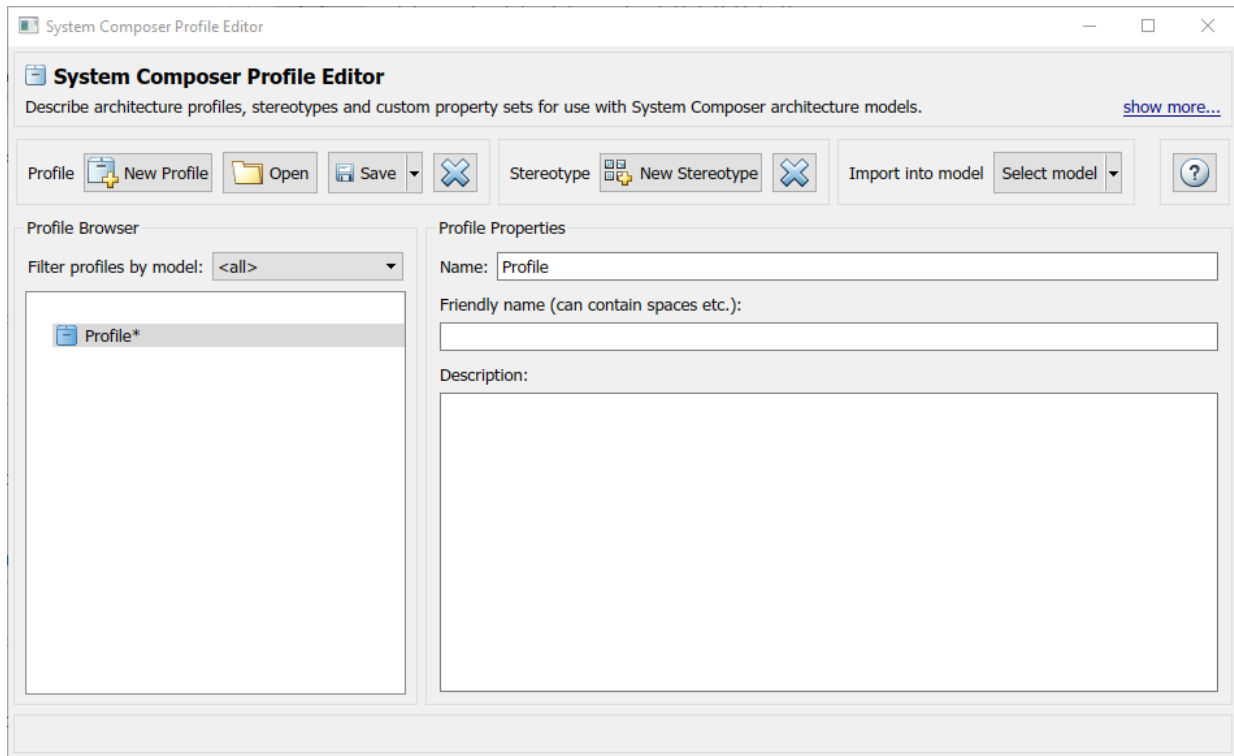
Create a Profile and Add Stereotypes

Create a profile to define a set of component, port, and connection types to be used in an architecture model. For example, a profile for an electromechanical system, such as a robot, could consist of these types:

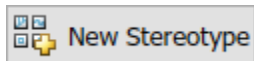
- Component types:
 - Electrical component
 - Mechanical component
 - Software component
- Connection types:
 - Analog signal connection
 - Data connection
- Port types
 - Data port


Define a profile using the Profile Editor. In any architecture model, select **Architecture >**

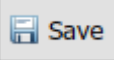
Profile >Profile Editor. Click . Select the new profile to start editing.



Name the profile and provide a description. Add stereotypes by clicking



. You can delete stereotypes and profiles by clicking  in their respective menus.

Save the profile with . The file name is the same as the profile name.

Add Properties with Stereotypes

Select a stereotype in a profile to define it:

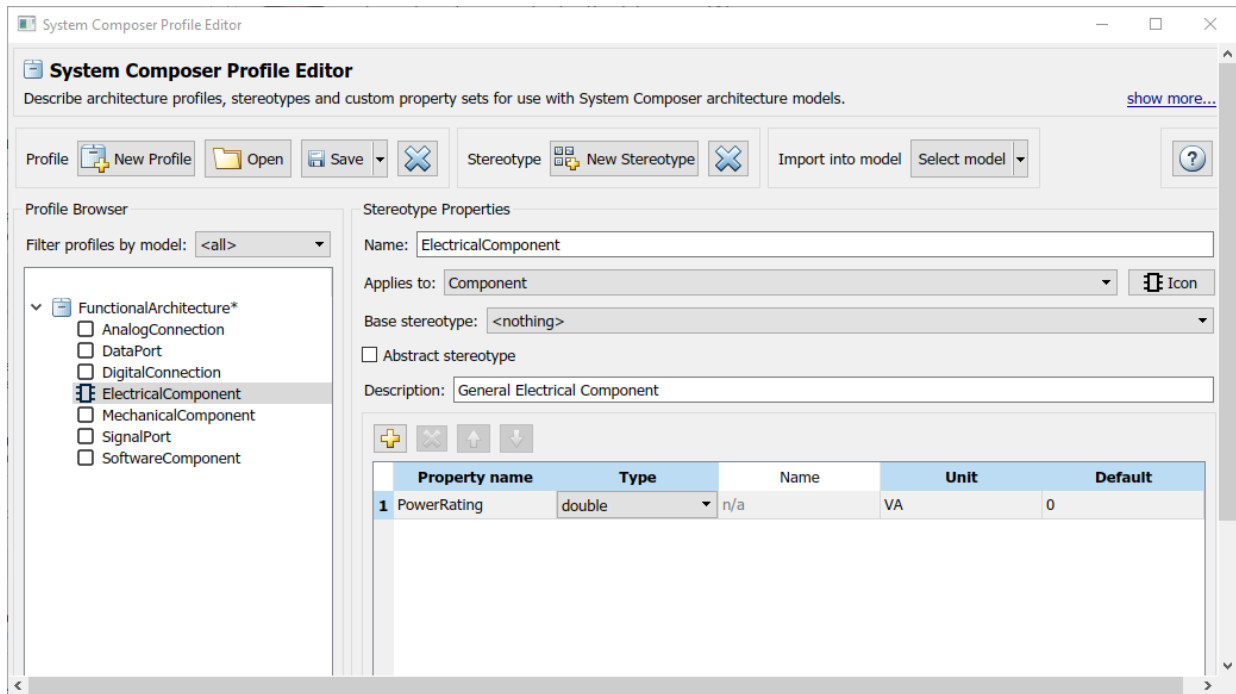
- **Name** — The name of the component type, for example, `ElectricalComponent`.


- **Applies to** — The model element type to which the stereotype applies. This field can be an architecture, component, port, or connector. You can apply this stereotype only to a model element of this type.
- **Icon** — Icon to be shown on the model element.
- **Base stereotype** — Other stereotype on which this stereotype is based. This can be empty.
- **Abstract stereotype** — A stereotype that is not intended to be applied directly to a model element. You can use abstract stereotypes only as the base stereotype for other stereotypes.

Add properties to a stereotype using . Define these fields for each property:

- Property name — valid variable name
- Type — numerical, string, or enumeration data type
- Unit — Value units as a string
- Default — Default value

4 Define Architectural Properties



Add, delete, and reorder properties using the property toolbar: 

You can create a stereotype that applies all model element types by setting the **Applies to** field to **<nothing>**. With these stereotypes, you can add properties to elements regardless of whether they are components, ports, connectors, or architectures.

Stereotype Properties

Name:

Applies to: Icon

Base stereotype:

Abstract stereotype

Description:

| | Property name | Type | Name | Unit | Default |
|---|---------------|-----------------------------------|------|------|---------|
| 1 | RefNumber | <input type="text" value="int8"/> | n/a | | 1 |

See Also

“Use Stereotypes and Profiles” on page 4-8

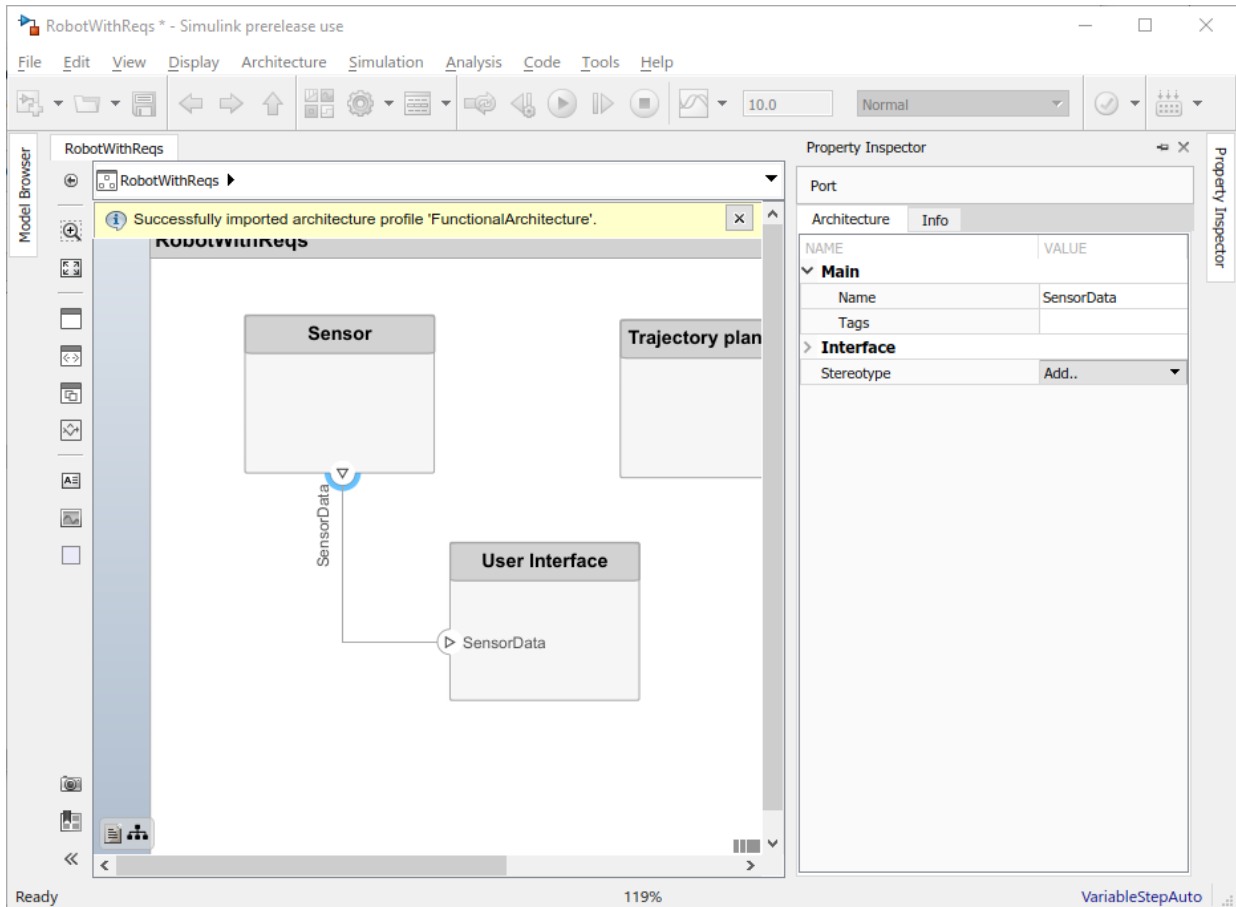
Use Stereotypes and Profiles

Use profiles to add properties to components, ports, and connectors. Import an existing profile, apply stereotypes, and add property values. To create a profile, see “Define Profiles and Stereotypes” on page 4-2.

Apply a Stereotype

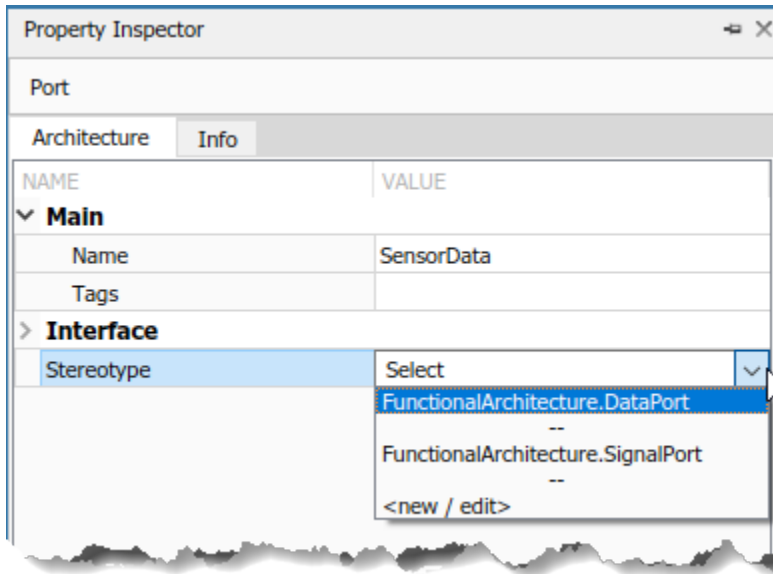
The Profile Editor is independent from the model that opens it, that is, you must explicitly import a new profile into a model. Select **Architecture > Profile > Import Profile** and select the profile to import. An architecture model can use multiple profiles at once.

Once the profile is available in the model, open the Property Inspector with **View > Property Inspector**. Select the model element.

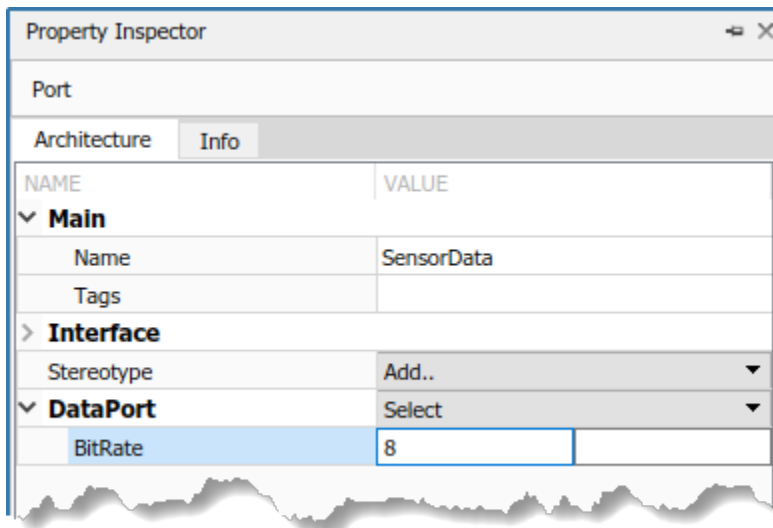


In the **Stereotype** field, use the drop-down to select the stereotype. Only the stereotypes that apply to this element type (for this example, a port) are available for selection. If no stereotype exists, you can use the **<new/edit>** option to open the profile editor and create one.

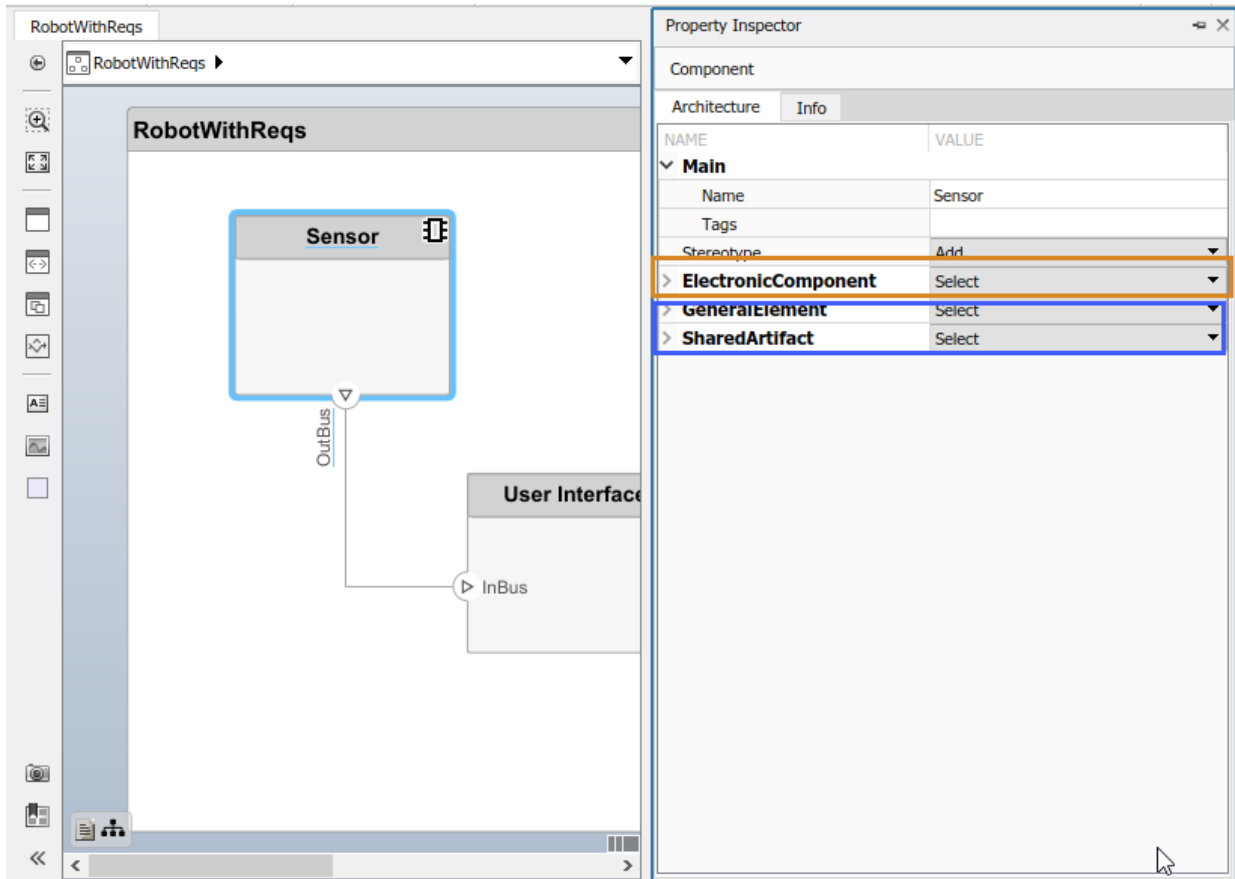
4 Define Architectural Properties



When you apply a stereotype to an element, a new set of properties appear in the Property Inspector under the name of the stereotype. Expand this set to edit the properties.



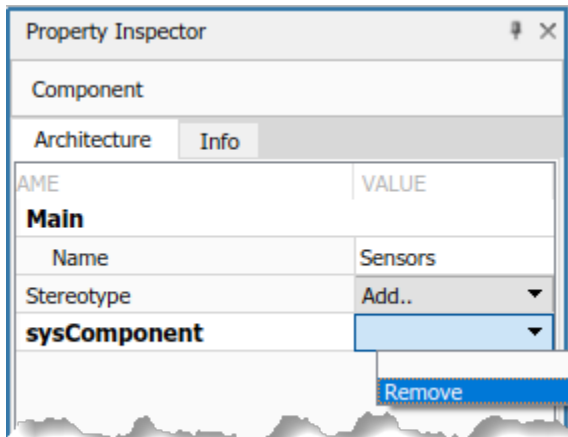
You can set multiple stereotypes for each element:



You can also apply component and connector stereotypes to all applicable elements at the same level. Navigate to the architecture and select **Architecture > Profile > Apply to all Components in this layer**, **Architecture > Profile > Apply to all Ports in this layer**, or **Architecture > Profile > Apply to all Connectors in this layer** appropriately.

Remove a Stereotype

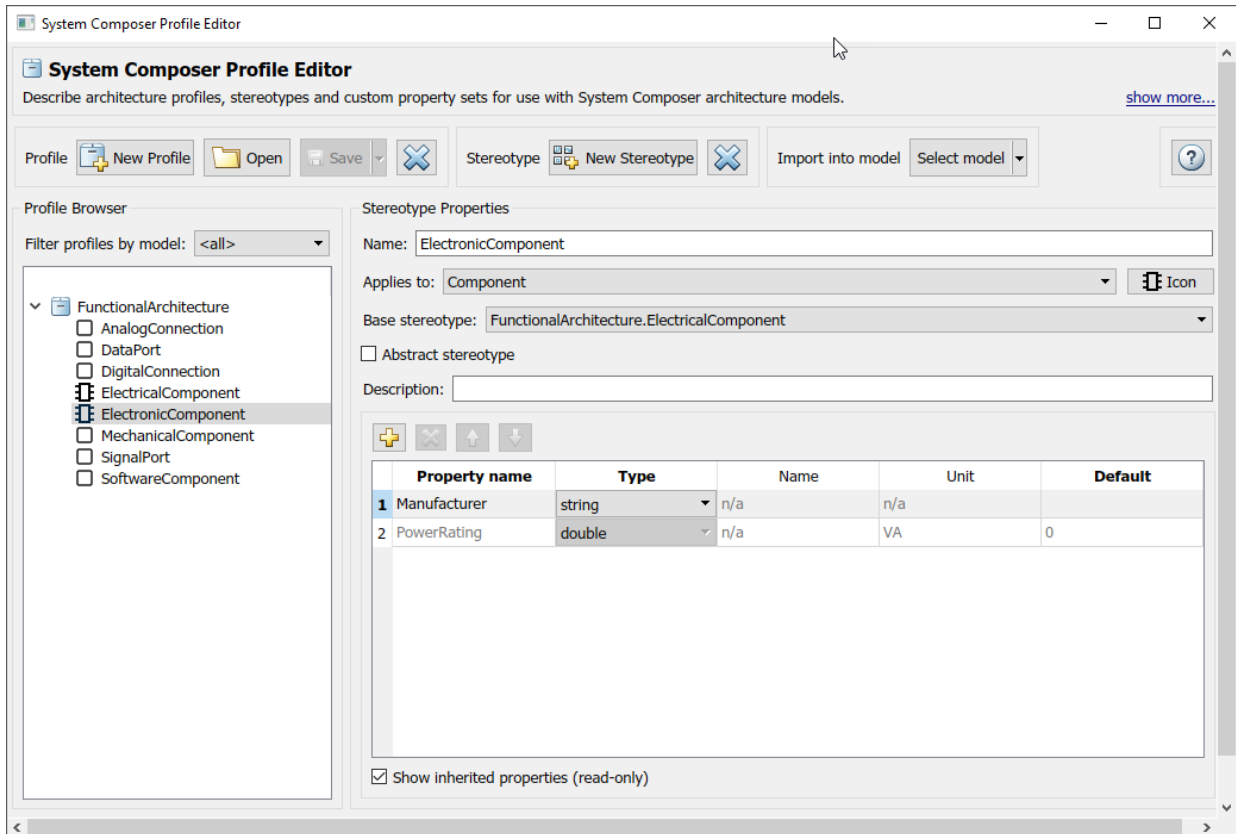
If a stereotype is no longer required for an element, remove it using the Property Inspector. Click **Select** next to the stereotype and choose **Remove**.



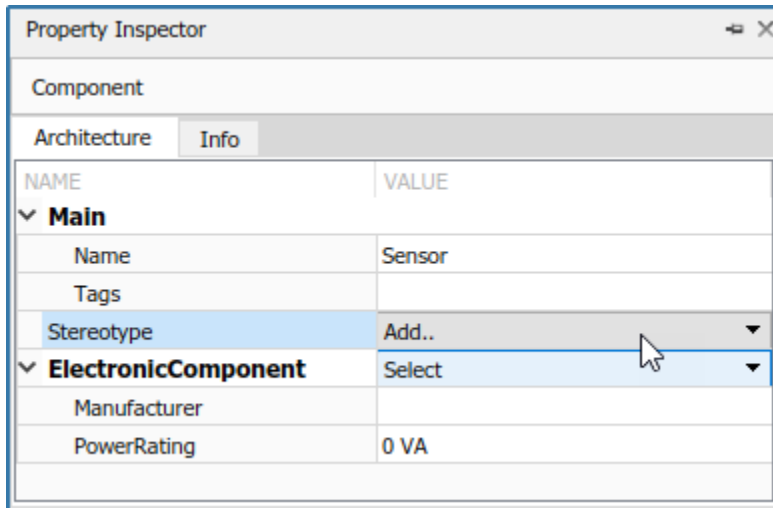
Extend a Stereotype

You can extend a stereotype by creating a new one based on the existing one. This allows you to control properties in a structural manner. For example, all components in a project may have a part number, only electrical components have a power rating, and only electronic components, which is a subset of electrical components, have manufacturer information. You can use an abstract stereotype to serve solely as a base for other stereotypes and not as a stereotype for any architecture model elements.

For example, create a new stereotype called `ElectronicComponent` in the Profile Editor. Select its base stereotype as `FunctionalArchitecture.ElectricalComponent`. Define properties that are in addition to those of the base stereotype. Check **Show inherited properties** at the bottom of the property list to show the properties of the base stereotype. You can edit only the properties of the selected stereotype; not the base stereotype.



When you apply the new stereotype, it carries its defined properties in addition to those of its base stereotype.



See Also

More About

- “Define Profiles and Stereotypes” on page 4-2
- “Analyze Architecture” on page 6-2

Use Simulink Models with System Composer

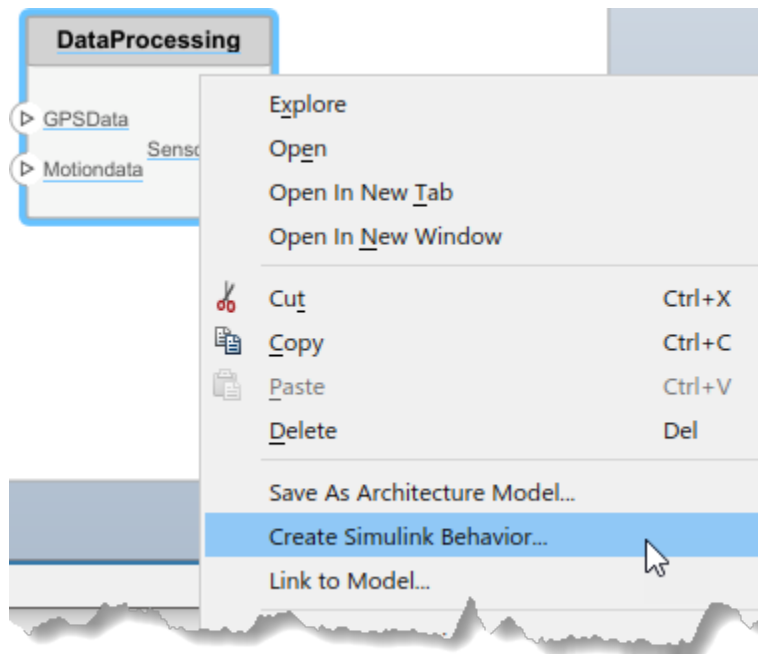
- “Implement Components in Simulink” on page 5-2
- “Extract Architecture from Simulink Model” on page 5-6

Implement Components in Simulink

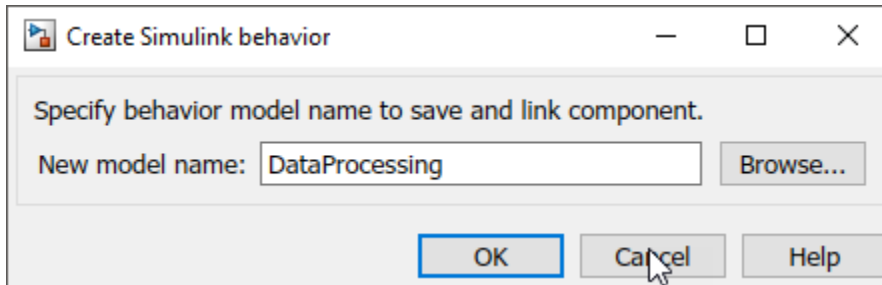
System design and architecture definition can involve behavior definition for some components, such as the algorithm to be used for a data processing component. Components in System Composer architecture models can have behavior definition Simulink models. Specify behavior for components by linking components to Simulink models.

Create a Simulink Behavior Model

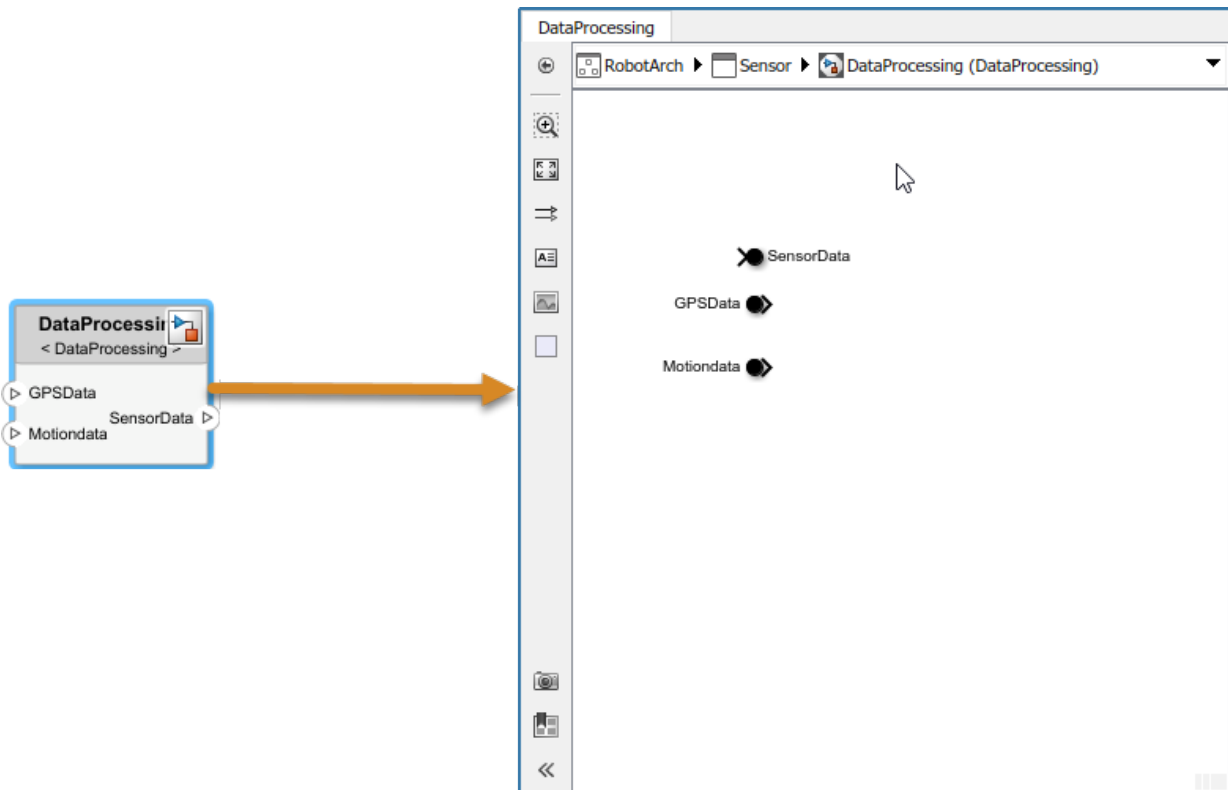
When a component does not require further decomposition from an architecture standpoint, you can design its behavior in Simulink. Right-click the component and select **Create Simulink Behavior**.



Provide a model name. The default name is the name of the component.

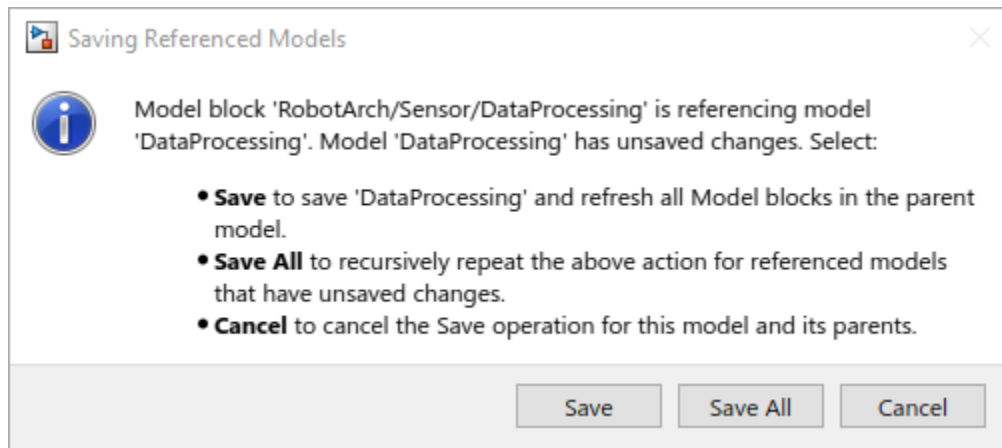


- A new Simulink model with the provided name is created. The root level ports of the Simulink model reflect the ports of the component.
- The component in the architecture model is linked to the Simulink model. The Simulink icon on the component indicates this is a Simulink link.



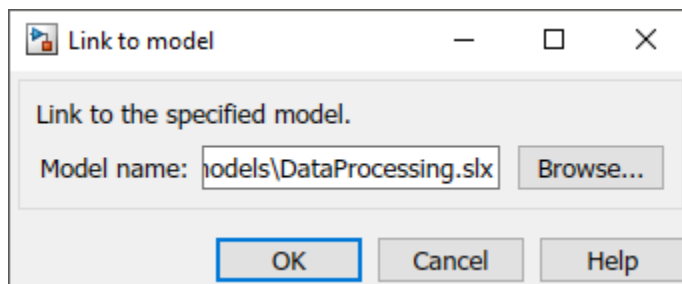
You can now go on to providing specific dynamics and algorithms in the referenced Simulink model. Adding root-level ports in the Simulink model creates additional ports on the referencing component.

You can access and edit a referenced Simulink model by double-clicking the component in the architecture model. When you save the architecture model, all unsaved Simulink behavior models it references must also be saved, and all linked components updated.



Link to an Existing Simulink Behavior Model

You can link to an existing Simulink behavior model from a System Composer component, provided that the component is not already linked to a reference architecture. Right-click the component and select **Link to Model**. Type in or browse for the name of a Simulink model.



Any subcomponents and ports that are present in the components get deleted when the component links to a Simulink model.

See Also

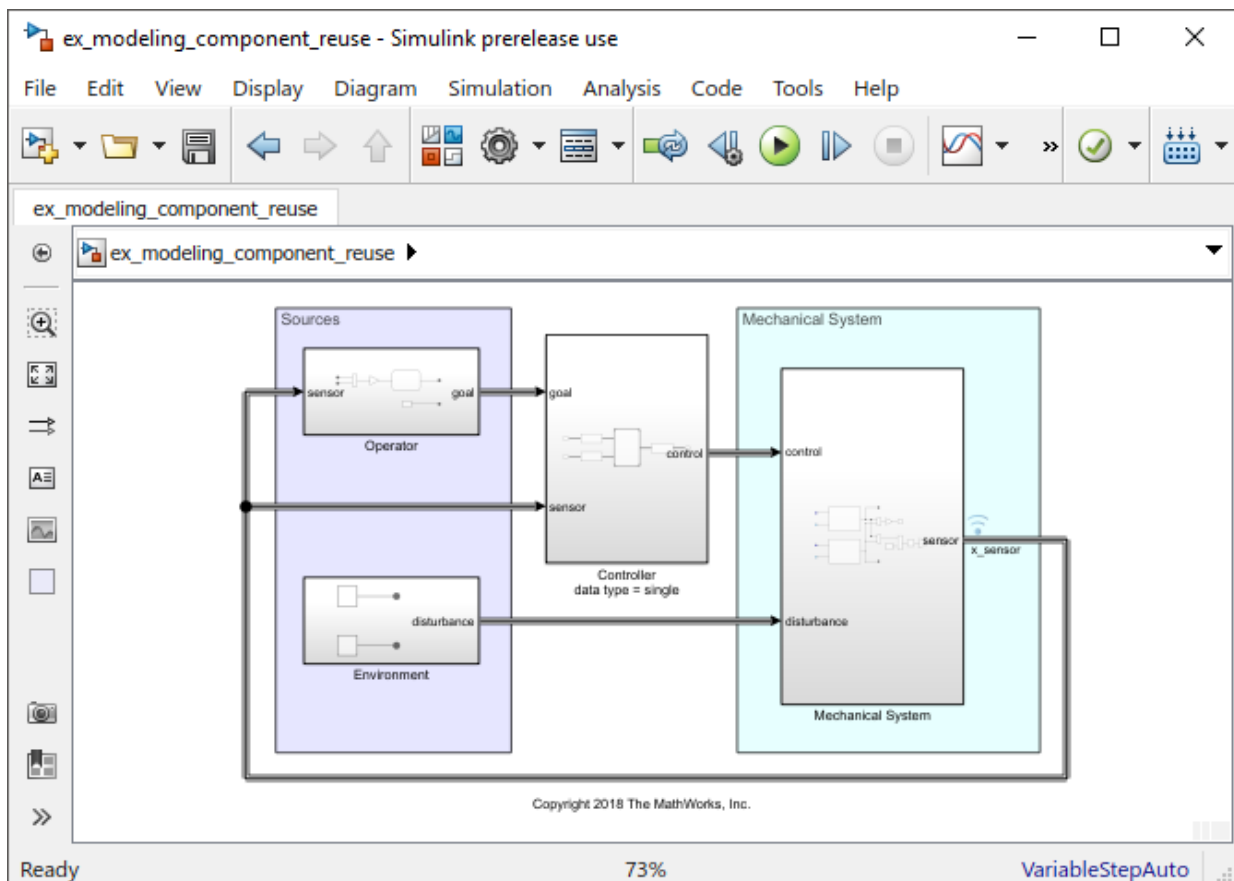
More About

- “Decompose and Reuse Components” on page 1-18
- “Extract Architecture from Simulink Model” on page 5-6

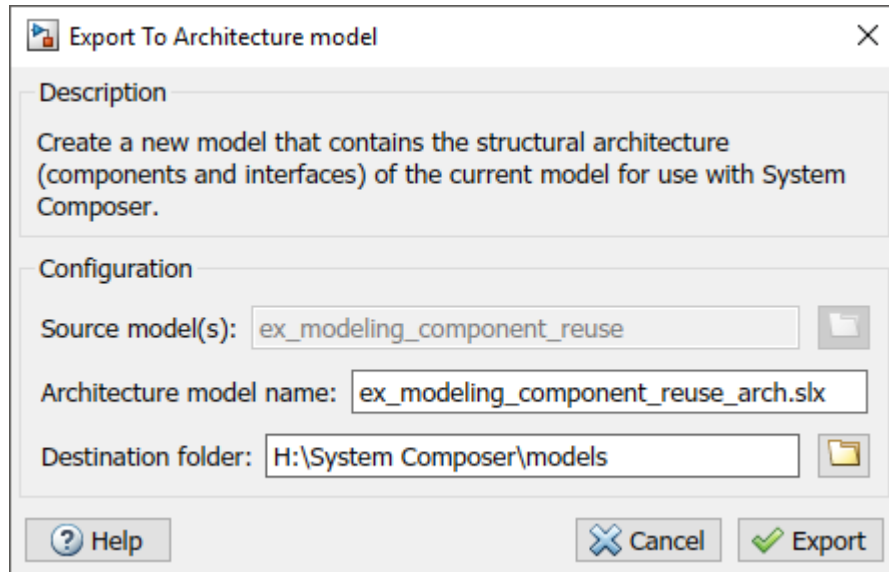
Extract Architecture from Simulink Model

You can use System Composer architecture editing and analysis capabilities on Simulink models by extracting the architecture from Simulink models. Model and Subsystem blocks, as well as all ports in a Simulink model represent architectural constructs; whereas all other blocks represent some kind of dynamic or algorithmic behavior. In the resulting architecture model you can choose to represent only architectural constructs, or link to behavior models.

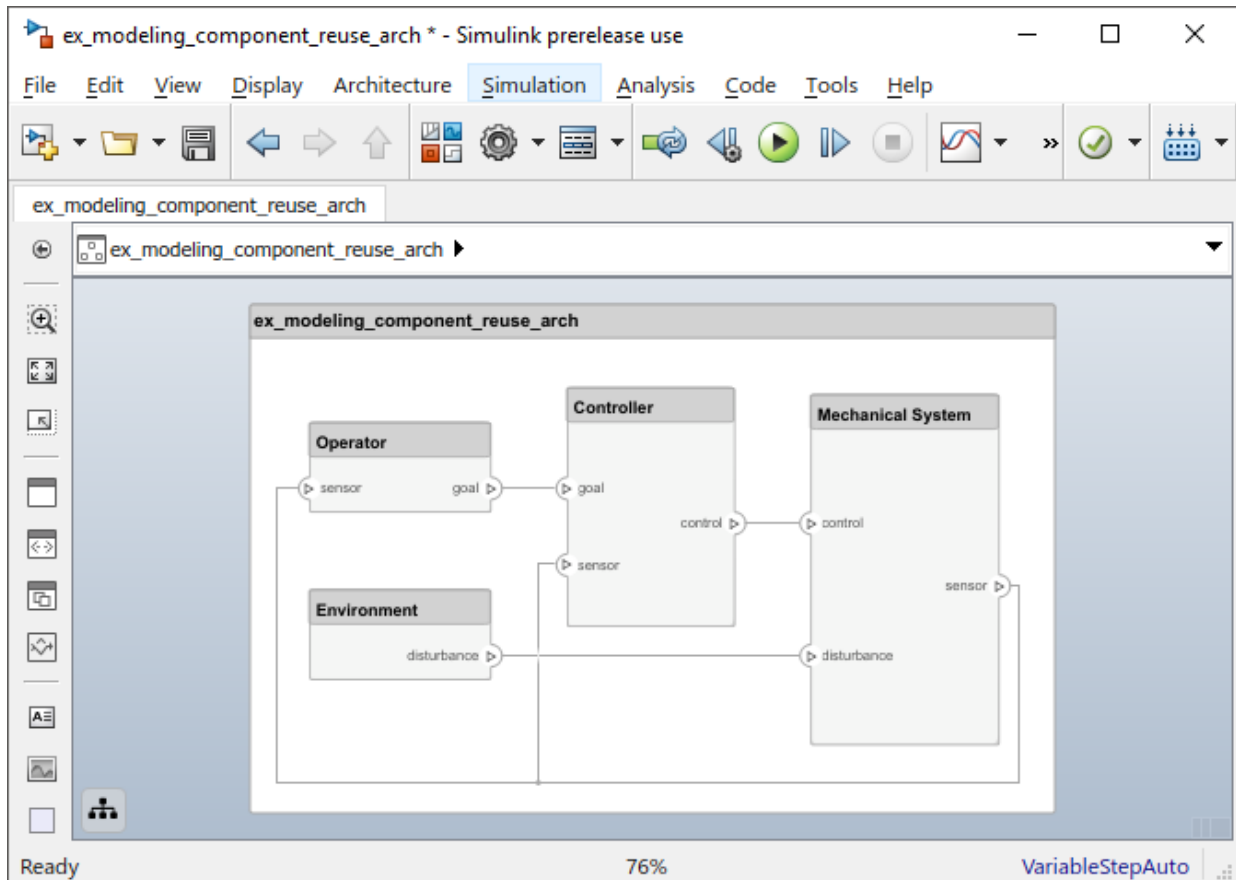
For example, open the model `ex_modeling_component_reuse.slx`.



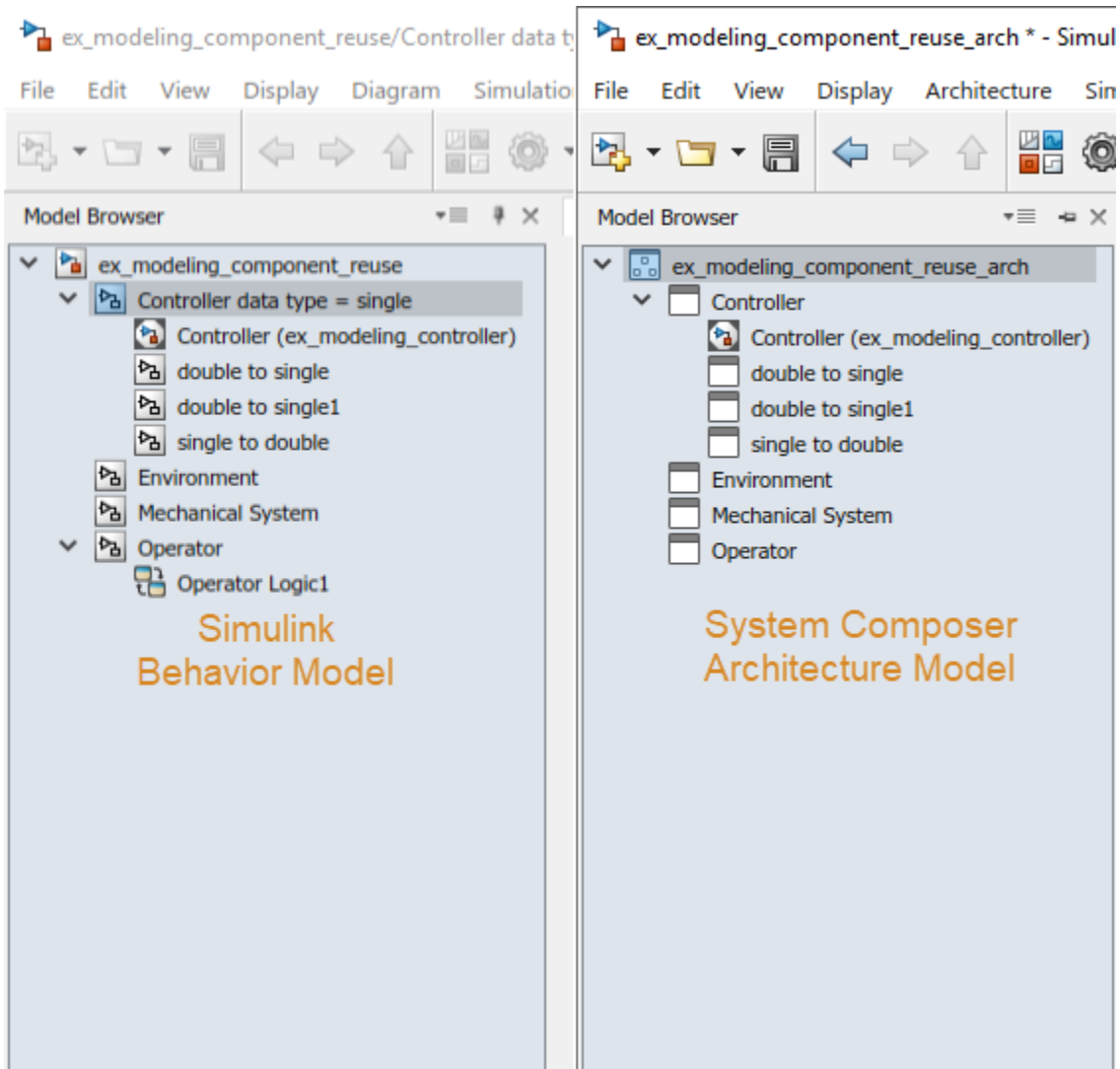
- 1 On the Simulink model, select **File > Export Model to > Architecture Model**.
- 2 Provide a name and path for the architecture model.



- 3 Click **Export**.

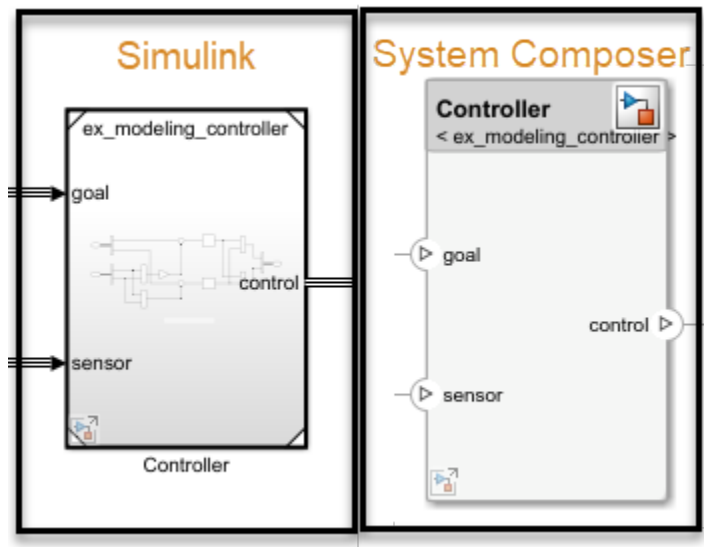


Each subsystem in the Simulink model corresponds to a component in the architecture model so that the hierarchy in the architecture model reflects the hierarchy of the behavior model.



The requirements for subsystems and Model blocks in the Simulink model are preserved in the architecture model.

Any Model block in the Simulink model that references another model corresponds to a component that links to that same referenced model.



Buses at subsystem and Model block ports, as well as their dictionary links are preserved in the architecture model.

You can use the exported model to add architecture-related information such as interface definitions, non-functional properties for model elements and analyze the design.

See Also

More About

- “Implement Components in Simulink” on page 5-2
- “Decompose and Reuse Components” on page 1-18

Analyze Architecture Model

Analyze Architecture

Write analyses based on element properties to perform data-driven trade studies and verify system requirements. Consider an electromechanical system where there is a trade-off between cost and weight, and lighter components tend to cost more. The decision process involves analyzing the overall cost and weight of the system based on the properties of its elements, and iterating on the properties to arrive at a solution that is acceptable both from the cost and weight perspective.

The analysis workflow consists of these steps:

- Define a profile containing a set of property sets that describe some analyzable properties (for example, cost and weight)
- Apply the profile to an architecture model and add property sets from that profile to elements of the model (components, ports, or connectors)
- Specify values for the properties on those elements
- Create an instance of the architecture model, which is a tree of elements, corresponding to the model hierarchy with all shared architectures expanded and a variant configuration applied
- Write an analysis function to compute values necessary for the study
- Run the analysis function

Set Tags and Properties for Analysis

Enable analysis by tagging model elements and setting property values. Load this model:

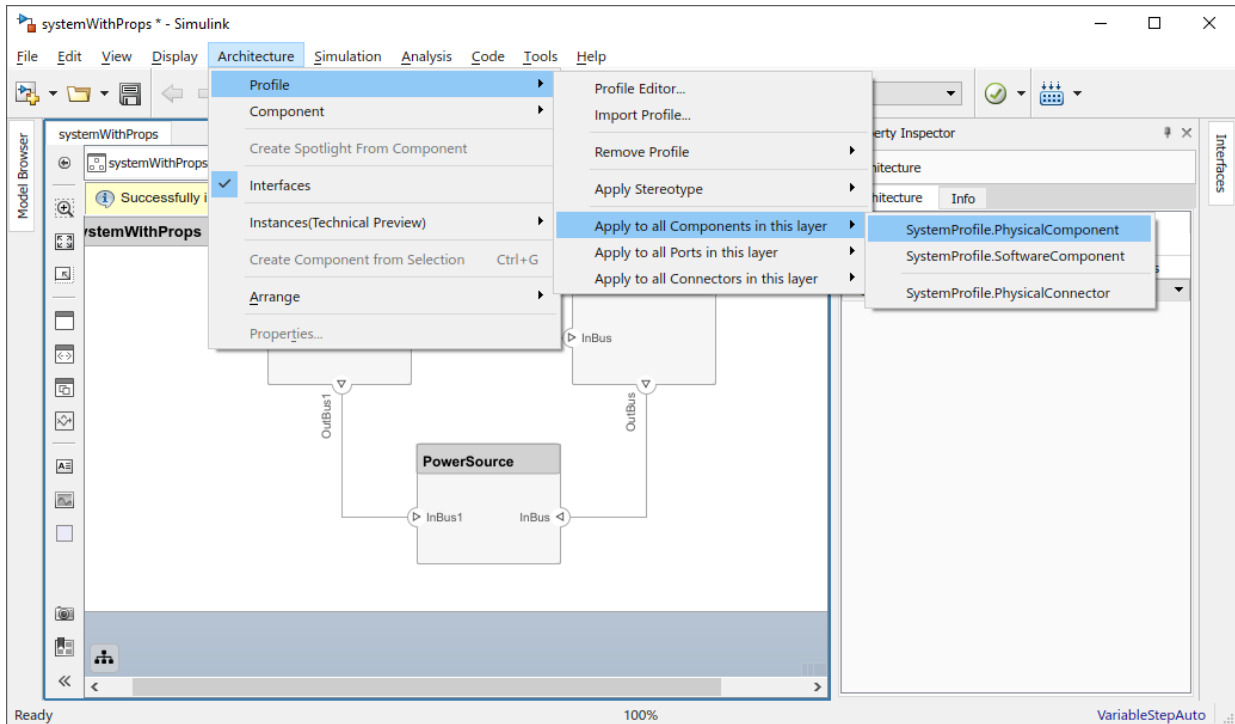
```
systemWithProps
```

Import a Profile

Enable analysis of properties by first importing a profile. On the model, select **Architecture > Profile > Import** and browse to the profile.

Apply Stereotypes to Model Elements

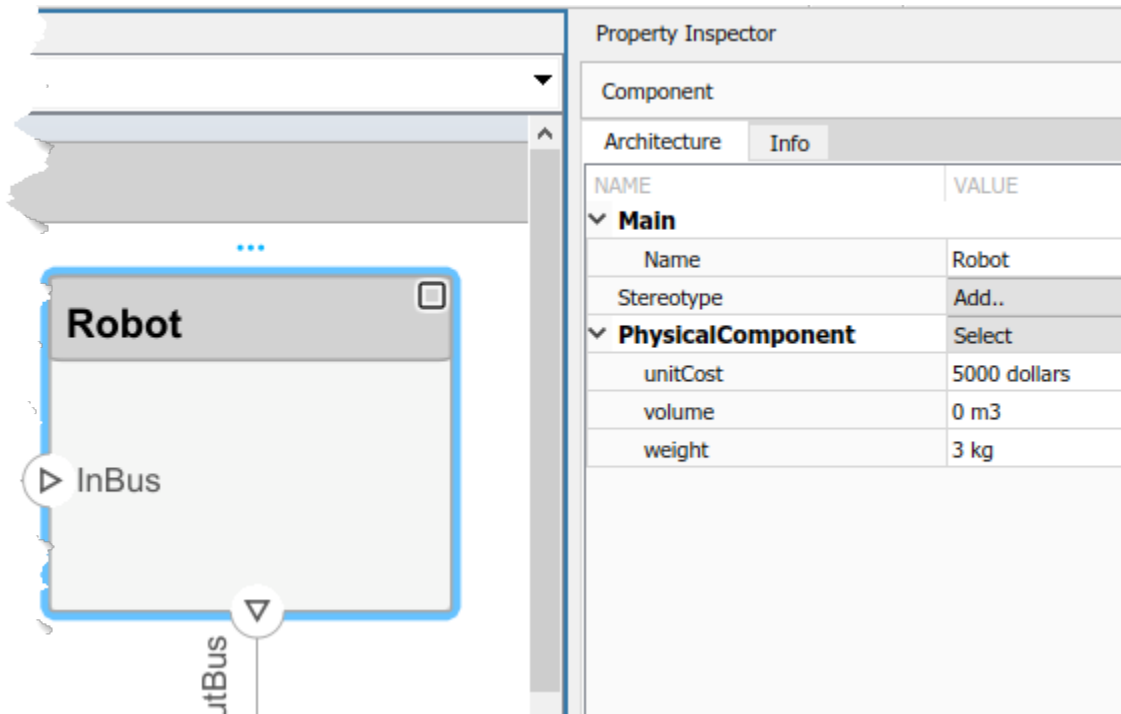
Apply stereotypes to all model elements that are part of the analysis. Use the menu items that apply stereotypes to all elements of a certain type, such as **Architecture > Profile > Apply to all Components in this layer** as necessary. Make sure you apply a stereotype to the top-level component, if a cumulative value is to be computed.



Set Property Values

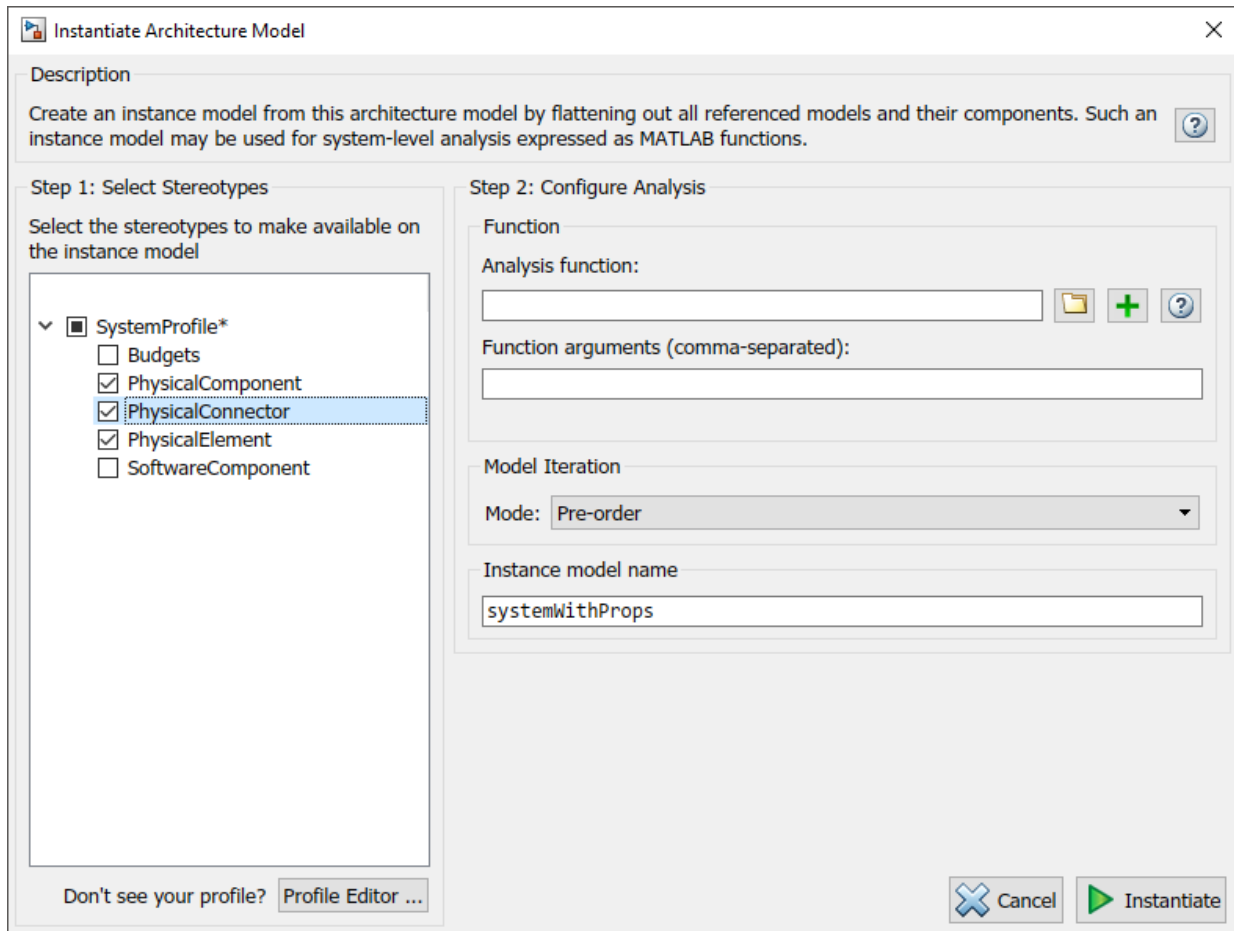
Set property values for each model element.

- 1 Open the Property Inspector.
- 2 Select the model element.
- 3 Expand the stereotype name and type values for properties.



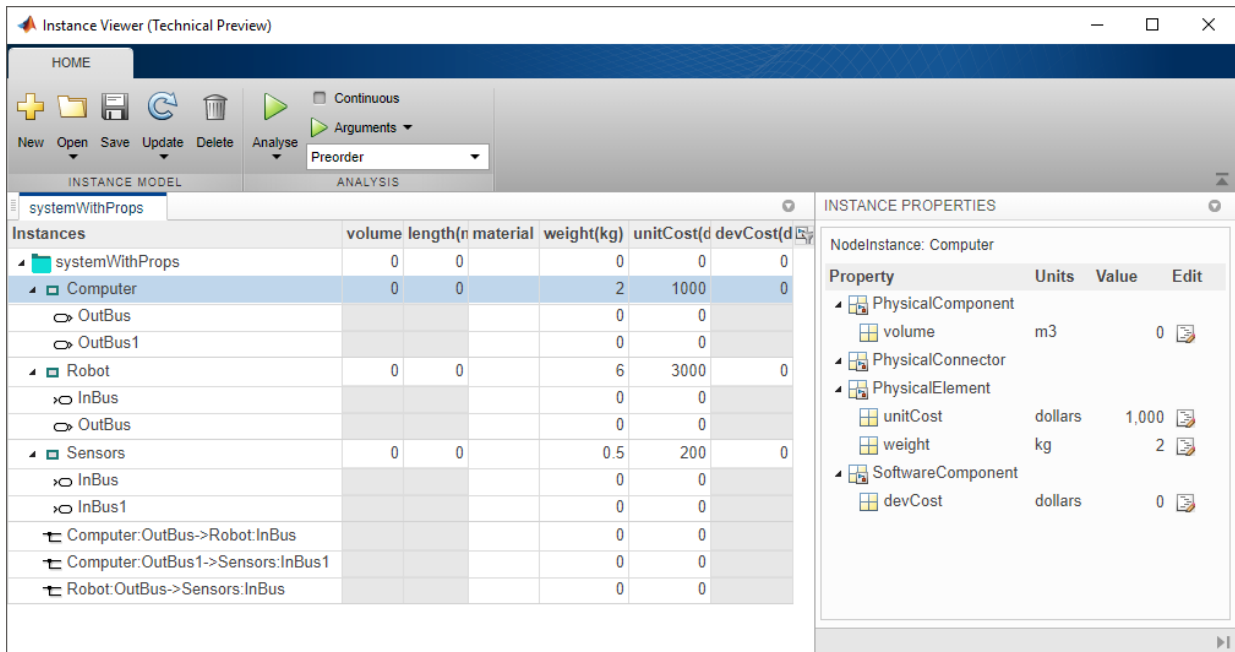
Create a Model Instance for Analysis

Create an instance of the architecture model that you can use for analysis. Select **Architecture > Create Analysis Model**. This dialog allows you to specify all the parameters required to create and view an analysis model.



The Stereotypes tree lists the stereotypes of all profiles that have been loaded in the current session and allows you to select those whose properties should be available in the instance model. You can browse for an analysis function, create a new one, or skip analysis at this point. If the analysis function requires inputs other than elements in the model (such as an exchange rate to compute cost) enter it in **Function arguments**. Select a mode for iterating through model elements, for example **Bottom-up** to move from the leaves of the tree to the root.


To view the instance, click **Instantiate**.



The Analysis Viewer shows all components, ports, and connectors in the first column. The other columns are properties for all stereotypes chosen for this instance. If a property is not part of a stereotype applied to an element, that field is greyed out. You can use the Filter button to hide properties for certain stereotypes. When you select an element, Instance Properties shows its stereotypes and property values. You can save an instance in a MAT-file, and open it again in the Analysis Viewer. If you make changes in the model while an instance is open, you can synchronize the instance with the model by clicking Update. Unsynchronized changes are shown in a different color.

Write Analysis Function

Write a function to analyze the architecture model using instance API. Analysis functions are MATLAB functions that compute values necessary to evaluate the architecture using properties of each element in the model instance.

You can add an analysis function as you set up the analysis instance. After you select the stereotypes of interest, create a template function by clicking the  button next to the **Analysis function** field. The generated M-file includes the code to obtain all property

values from all stereotypes that are subject to analysis. The analysis function operates on a single element — aggregate values are generated by iterating this function over all elements in the model when you run the analysis from Analysis Viewer.

```
function ex_zcRobot_props_analysis(instance,varargin)
% ex_zcRobot_props_analysis Example Analysis Function

if instance.isComponent()
    sysComponent_unitPrice = instance.getValue("sysComponent.unitPrice");
    for child = instance.Components
        comp_price = child.getValue("sysComponent.unitPrice");
        sysComponent_unitPrice = sysComponent_unitPrice + comp_price;
    end
    for child = instance.Connectors
        unitPrice = child.getValue("sysConnector.unitPrice");
        length = child.getValue("sysConnector.length");
        sysComponent_unitPrice = unitPrice*length + comp_price;
    end
    instance.setValue("sysComponent.unitPrice",sysComponent_unitPrice)
end
```

In the generated file, `instance` is the instance of the element on which the analysis function runs currently. You can perform these operations for analysis:

- Access a property of the instance:
`instance.getValue("<stereotype>.<property>")`
- Set a property of an instance:
`instance.setValue("<stereotype>.<property>", value)`
- Access the subcomponents of a component: `instance.Components`
- Access the connectors in component: `instance.Connectors`

The `getValue` function generates an error if the property does not exist. You must use error handling functions such as `try-catch` statements if it is possible that some elements in the model do not use the stereotypes.

As an example, this code computes the weight of a component as a sum of the weights of its subcomponents.

```
if instance.isComponent()
    weight = 0;
    for child=instance.Components
        subcomp_weight = child.getValue("PhysicalElement.weight");
        weight = weight + subcomp_weight;
    end
end
```

```
end
instance.setValue("PhysicalElement.weight",weight)
end
```

Once the analysis function is complete, add it to the analysis. An analysis function can take additional input arguments, for example, a conversion constant if the weights are in different units in different stereotypes. When this code runs for all components recursively, starting from the deepest components in the hierarchy to the top level, the overall weight of the system is assigned to the `weight` property of the top-level component.

Run Analysis Function

Run an analysis function using the Analysis Viewer.

- 1 Select or change the analysis function using the **Analyze** menu.
- 2 Select the iteration method.
 - **Preorder**: Start from the top level, move to a child component, process the subcomponents of that component recursively before moving to a sibling component.
 - **Topdown**: Like pre-order, but process all sibling components before moving to their subcomponents.
 - **Postorder**: Start from components with no subcomponents, process each sibling and then move to parent.
 - **Bottomup**: Like post-order, but process all subcomponents at the same depth before moving to their parents.

The iteration method depends on what kind of analysis is to be run. For example, for an analysis where the component weight is the sum of the weights of its components, you must make sure the subcomponent weights are computed first, so the iteration method must be bottom-up.

- 3 Click the analyze button.

System Composer runs the analysis function over each model element and computes results. The computed properties are shown in a different color in the Analysis Viewer.

The screenshot shows a software interface with a ribbon menu and a table. The ribbon menu includes sections for 'INSTANCE MODEL', 'ANALYSIS', and 'UPDATE'. The 'ANALYSIS' section has a dropdown menu set to 'BottomUp'. The table below has columns for 'Instances', 'cost', 'totalCost', and 'price'. The 'totalCost' column is highlighted in yellow.

| Instances | cost | totalCost | price |
|---|------|-----------|-------|
| <ul style="list-style-type: none"> ▲ NewModel5 ▲ Component ▲ Component1 Component:OutBus->Component1:InBus | 0 | 16 | |
| | 5 | 5 | |
| | 10 | 10 | |
| | | | 1 |

See Also

`systemcomposer.analysis.Instance`

More About

- “Define Profiles and Stereotypes” on page 4-2
- “Use Stereotypes and Profiles” on page 4-8

