

# System Composer™

## Getting Started Guide



# MATLAB® & SIMULINK®

R2021b



## How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



Phone: 508-647-7000



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

### *System Composer™ Getting Started Guide*

© COPYRIGHT 2019–2021 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](http://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](http://www.mathworks.com/patents) for more information.

### **Revision History**

March 2019	Online only	New for Version 1.0 (Release 2019a)
September 2019	Online only	Revised for Version 1.1 (Release 2019b)
March 2020	Online only	Revised for Version 1.2 (Release 2020a)
September 2020	Online only	Revised for Version 1.3 (Release 2020b)
March 2021	Online only	Revised for Version 2.0 (Release 2021a)
September 2021	Online only	Revised for Version 2.1 (Release 2021b)

## Product Overview

### 1

<b>System Composer Product Description</b> .....	<b>1-2</b>
--	------------

## Compose an Architecture Model

### 2

<b>Compose and Analyze a System Using an Architecture Model</b> .....	<b>2-2</b>
---	------------

#### **Create an Architecture Model with Interfaces and Requirement Links**

.....	<b>2-4</b>
Robot Arm Architecture Model .....	<b>2-4</b>
Visually Represent the System .....	<b>2-5</b>
Edit Data Interfaces .....	<b>2-13</b>
Decompose Components .....	<b>2-15</b>
Manage Requirement Links .....	<b>2-16</b>

#### **Extend Architectural Design Using Stereotypes** .....

Mobile Robot Architecture Model .....	<b>2-18</b>
Load Architecture Model Profile .....	<b>2-19</b>
Apply Stereotypes to Model Elements .....	<b>2-21</b>
Set Properties .....	<b>2-23</b>
Mobile Robot Architecture Model with Properties .....	<b>2-25</b>

#### **Analyze an Architecture Model with an Analysis Function** .....

Mobile Robot Architecture Model with Properties .....	<b>2-27</b>
Perform an Analysis .....	<b>2-28</b>

#### **Inspect Components in Custom Architecture Views** .....

Mobile Robot Architecture Model with Properties .....	<b>2-31</b>
Inspect the Component and Its Connectivity .....	<b>2-32</b>
Create a Filtered Architecture View .....	<b>2-34</b>

#### **Implement Behaviors for Architecture Model Simulation** .....

Robot Arm Architecture Model .....	<b>2-37</b>
Reference Simulink Behavior Model in Component .....	<b>2-38</b>
Add Stateflow Chart Behavior to Component .....	<b>2-40</b>
Design Software Architecture in Component .....	<b>2-41</b>
Represent System Interaction Using Sequence Diagrams .....	<b>2-43</b>

<b>System Composer Concepts</b> .....	<b>3-2</b>
Author Architecture Models .....	<b>3-2</b>
Manage Variants .....	<b>3-4</b>
Manage Interfaces .....	<b>3-5</b>
Author Physical Models .....	<b>3-8</b>
Extend Architectural Elements .....	<b>3-10</b>
Manage and Verify Requirements .....	<b>3-12</b>
Create Custom Views .....	<b>3-15</b>
Allocate Architecture Models .....	<b>3-17</b>
Analyze Architecture Models .....	<b>3-18</b>
Author Model Behavior .....	<b>3-20</b>
Design Software Architectures .....	<b>3-23</b>

# Product Overview

---

## **System Composer Product Description**

### **Design and analyze system and software architectures**

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

Architecture models consisting of components and interfaces can be authored directly, imported from other tools, or populated from the architectural elements of Simulink designs. You can describe your system using multiple architecture models and establish direct relationships between them via model-to-model allocations. Behaviors can be captured in sequence diagrams, state charts, or Simulink models. You can define and simulate the execution order of component functions and generate code from your software architecture models (with Simulink and Embedded Coder®).

To investigate specific design or analysis concerns, you can create custom live views of the model. Architecture models can be used to analyze requirements, capture properties via stereotyping, perform trade studies, and produce specifications and interface control documents (ICDs).

# Compose an Architecture Model

---

- “Compose and Analyze a System Using an Architecture Model” on page 2-2
- “Create an Architecture Model with Interfaces and Requirement Links” on page 2-4
- “Extend Architectural Design Using Stereotypes” on page 2-18
- “Analyze an Architecture Model with an Analysis Function” on page 2-27
- “Inspect Components in Custom Architecture Views” on page 2-31
- “Implement Behaviors for Architecture Model Simulation” on page 2-37

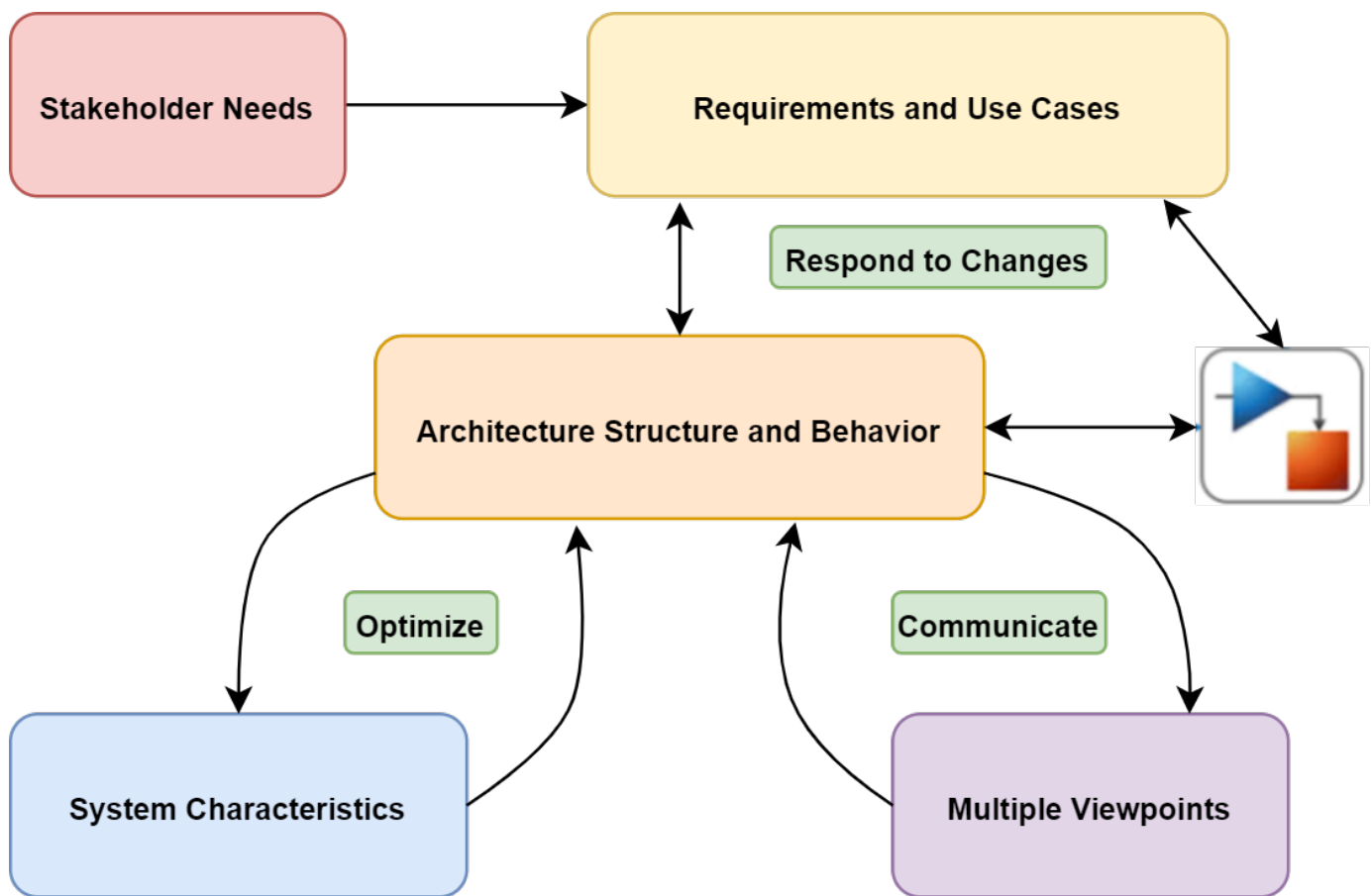
## Compose and Analyze a System Using an Architecture Model

A system refers to a composition of elements that interact to achieve a goal no single element could accomplish on its own. The constituent elements of a system can include mechanical parts, electrical circuits, computer hardware, and software. A system specification describes the system elements, their characteristics and properties, their interactions with each other, and the desired interaction (or interface) of the overall system with its environment.

System Composer enables you to describe systems in terms of architecture models as a combination of structural elements that are further elaborated by underlying behavioral descriptions. These descriptive models may sometimes be presented as distinct diagrams that are consistent with each other.

To perform a basic systems engineering workflow to design a mobile robotic arm using System Composer, see “Create an Architecture Model with Interfaces and Requirement Links” on page 2-4.

The model-based systems engineering (MBSE) workflow enabled by System Composer involves starting with stakeholder needs, identifying requirements and use cases, designing an architecture iteratively, and implementing functionality using behavior models. You can also use analyses and trade studies to optimize architectural design and communicate facets of the system using architecture views. This figure illustrates an MBSE workflow.



With System Composer, you can implement a systems engineering workflow:



**1 Author Architecture Models and Define System Requirements**

- Create hierarchical models of system structure that represent functional, logical, or physical decompositions of the system using components, ports, and connectors.
- Import models from MATLAB® tables and export them with System Composer changes.
- Create and manage data interfaces between structural architecture elements.
- Manage model-to-model allocations to establish relationships between software components and hardware components and to indicate deployment strategies.
- Refine and elaborate requirements using Simulink Requirements™. Link requirements to architectural model elements.

**2 Define Metadata, Analyze Architectures, and Generate Views**

- Extend base architectural elements to create domain-specific conceptual representations.
- Perform static analysis and trade studies to optimize system architectures.
- Filter views of the system structure using a component diagram, hierarchy diagram, or class diagram.

**3 Simulate System Behavior of Architecture Models and Verify Requirements**

- Specify component behavior using block diagrams in Simulink, state machines in Stateflow®, and physical interfaces in Simscape™ using subsystem component behaviors.
- Represent the interaction between structural elements of an architecture as a sequence of message exchanges with a sequence diagram in the Architecture Views Gallery.
- Design a software architecture model, define the execution order of the functions from the components, simulate the design at the architecture level, and generate code.
- Verify and validate requirements with Simulink Test™.

For definitions and applications of common System Composer terms and concepts, see “System Composer Concepts” on page 3-2.

**See Also****More About**

- “Create an Architecture Model with Interfaces and Requirement Links” on page 2-4

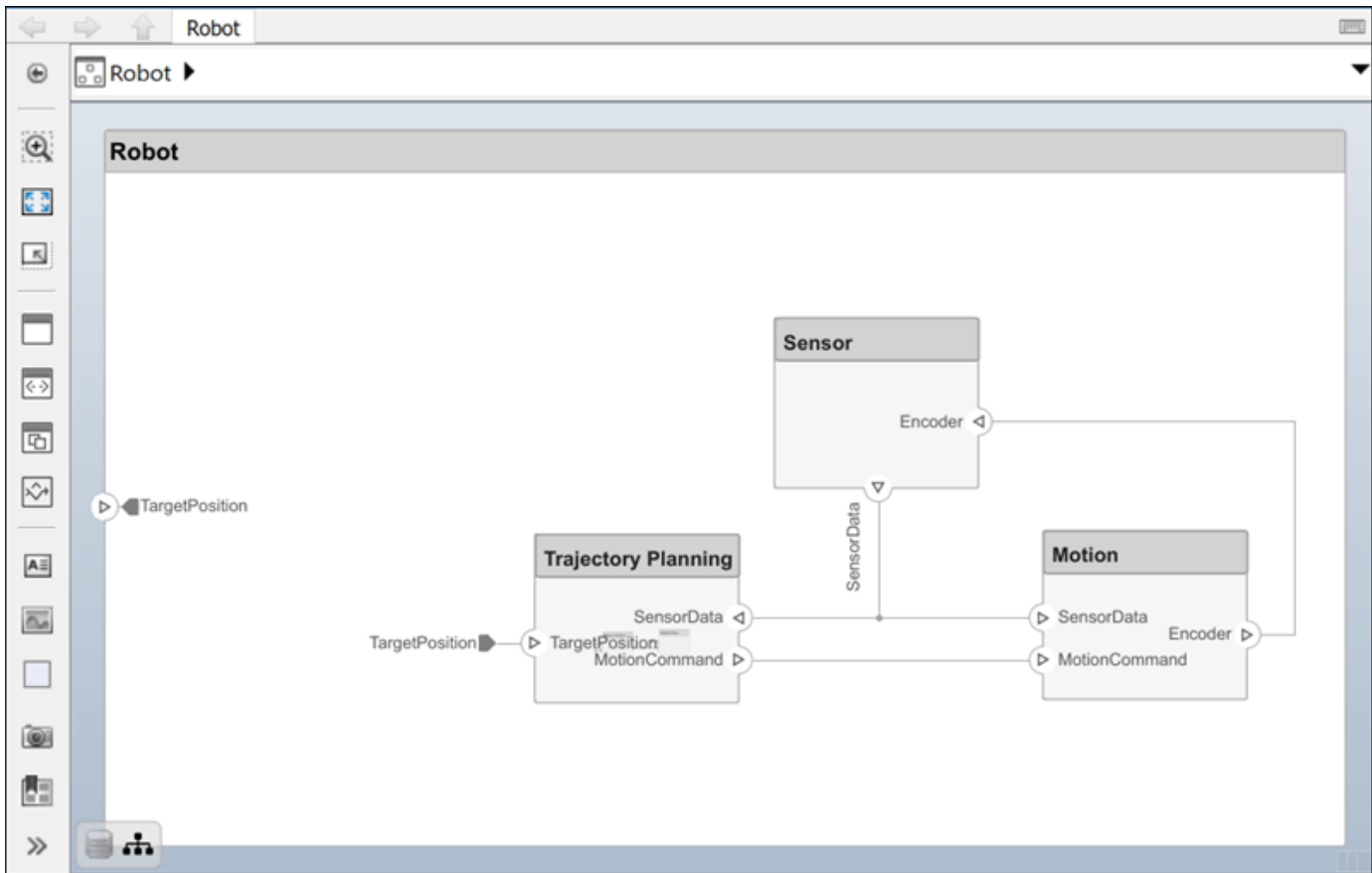
## Create an Architecture Model with Interfaces and Requirement Links

In this section...
“Robot Arm Architecture Model” on page 2-4
“Visually Represent the System” on page 2-5
“Edit Data Interfaces” on page 2-13
“Decompose Components” on page 2-15
“Manage Requirement Links” on page 2-16

Create an architecture model of a robot arm using System Composer. Define interfaces on ports and link requirements on components. A Simulink Requirements license is required to manage requirements. When you complete the steps, you will have created a completed model.

### Robot Arm Architecture Model

Open the architecture model of a robot arm that consists of sensors, motion actuators, and a planning algorithm. You can use System Composer to view the interfaces and manage the requirements for this model.



## Visually Represent the System

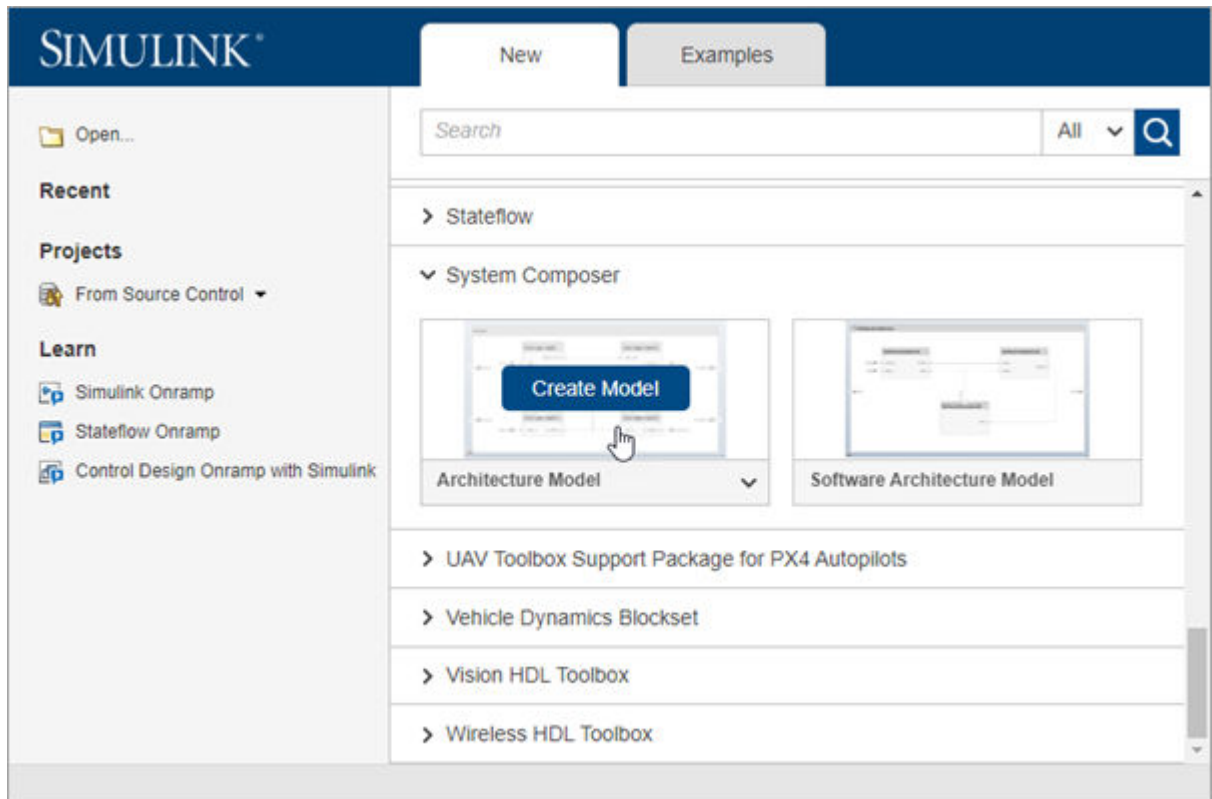
Implementing an architectural design starts with visually representing the system using components and their connections. Create an architecture model, represent the system components, and draw the connections between them.

### Create Architecture Model

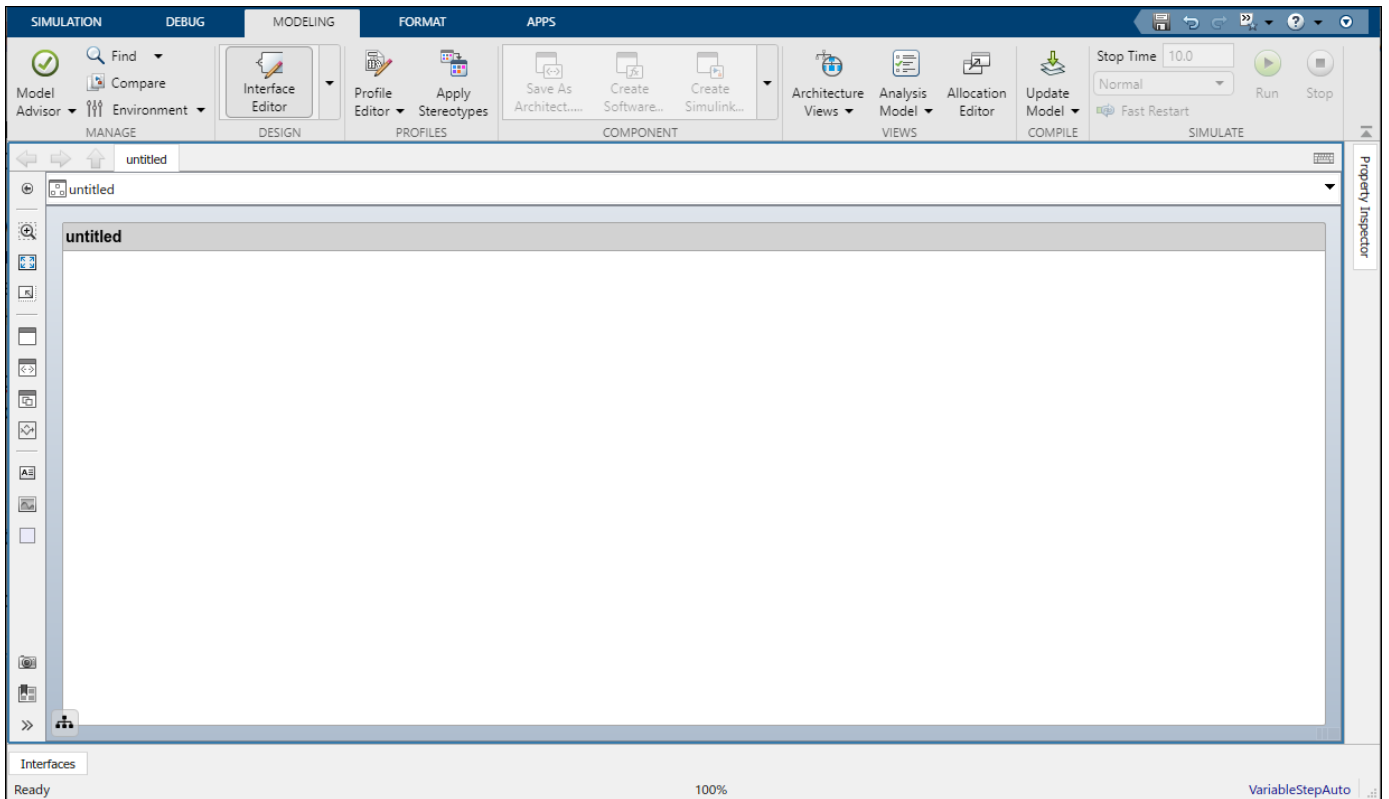
- 1 In the MATLAB Command Window, type:  

```
systemcomposer
```

The Simulink Start Page opens to System Composer.
- 2 Click **Architecture Model**.

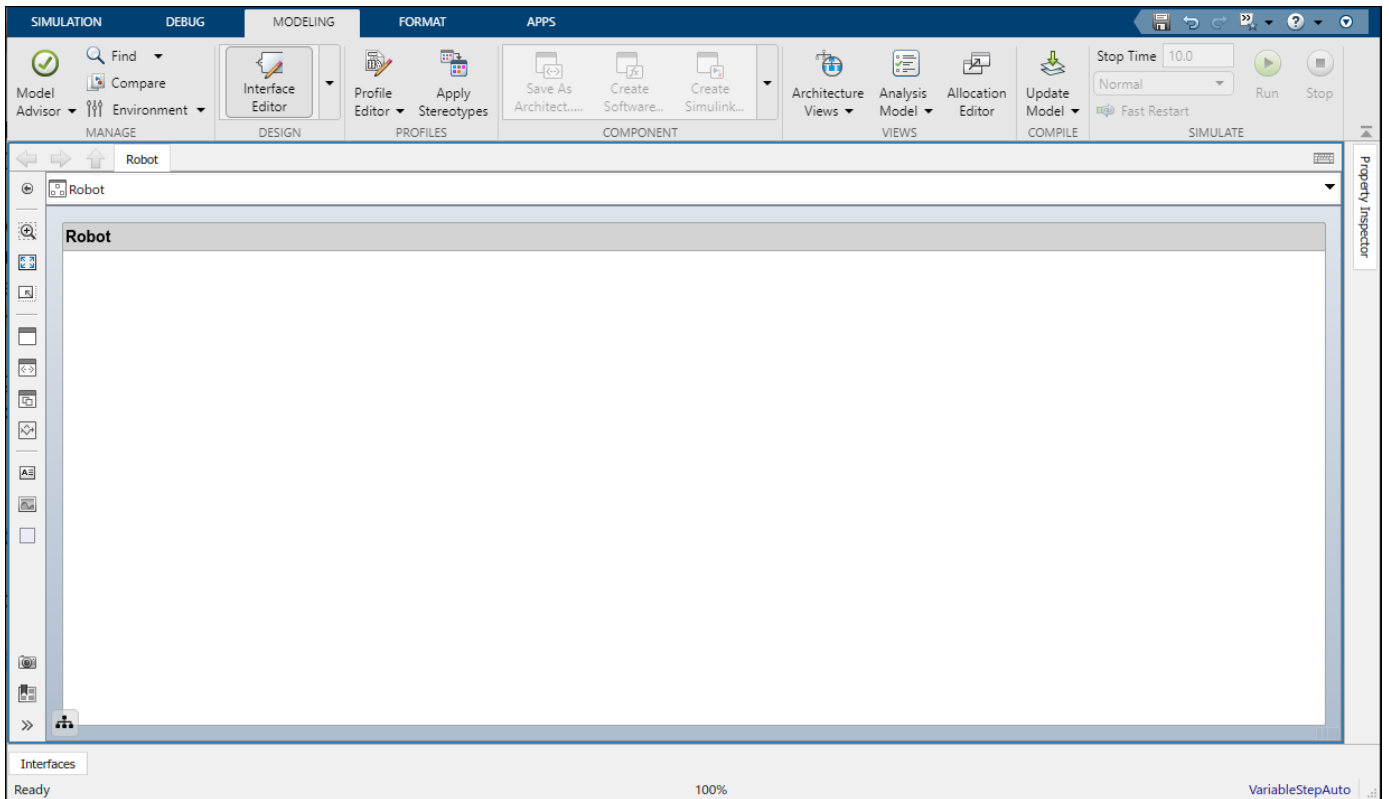


A new, blank architecture model canvas opens. You can identify an architecture model by the badge in the lower left corner and the component palette on the left side.



- 3 Double-click the architecture model header and change untitled to Robot. The name of the model generally reflects the system whose architecture you are building.

## 2 Compose an Architecture Model

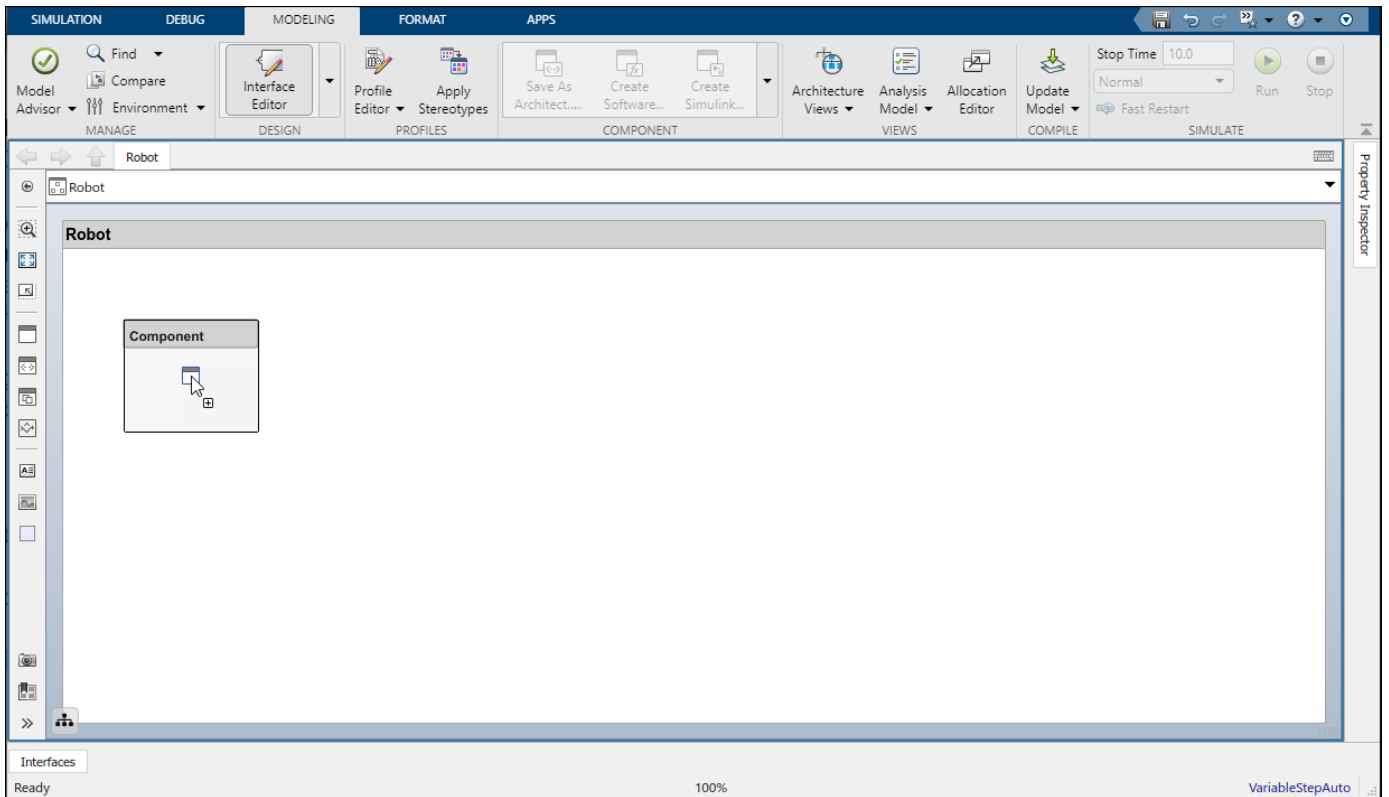


- 4 Save the model.

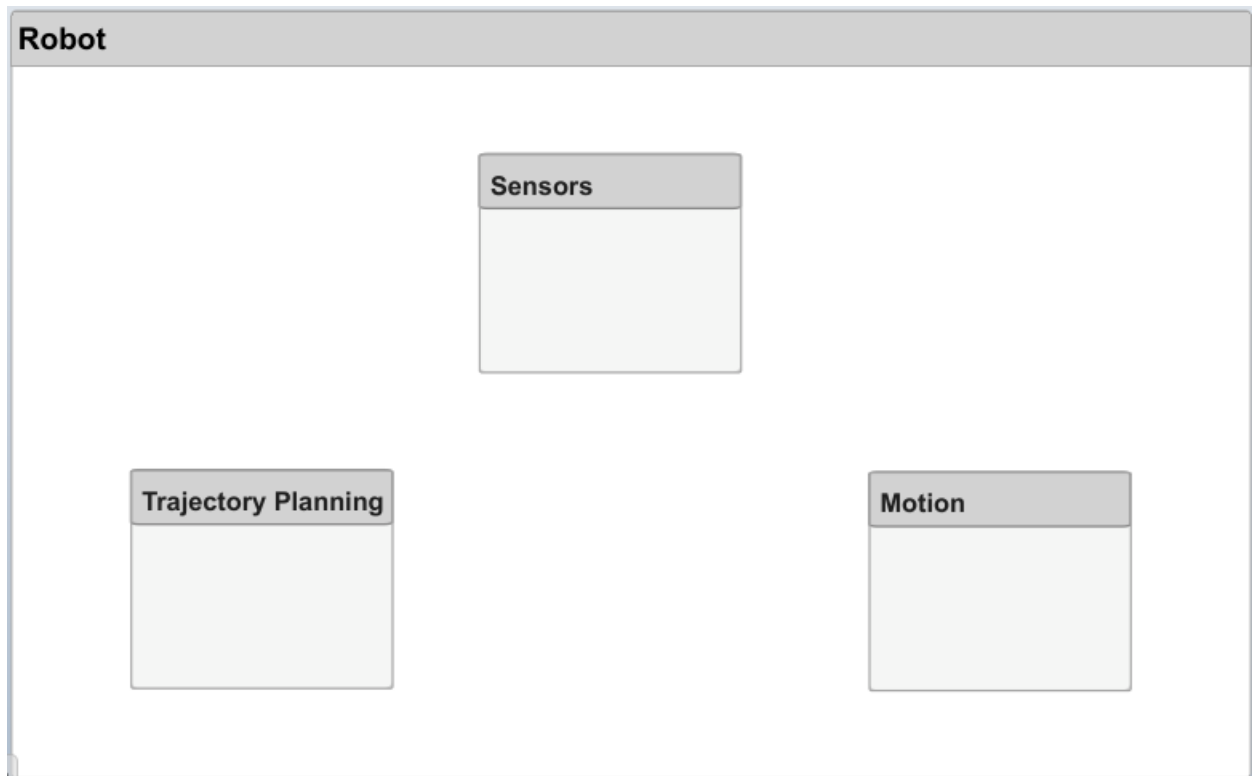
### Draw Components

Design a mobile robotic arm where a sensor senses position and trajectory planning computes a path to a location that the robot needs to reach using motion. An architecture model of such a system could consist of three primary components: **Sensors**, **Trajectory Planning**, and **Motion**. You can represent these components in System Composer using three Component blocks.

- 1 Click and drag a Component  from the left-side palette.



- 2 Rename the component as Sensors.
- 3 Add Trajectory Planning and Motion components.



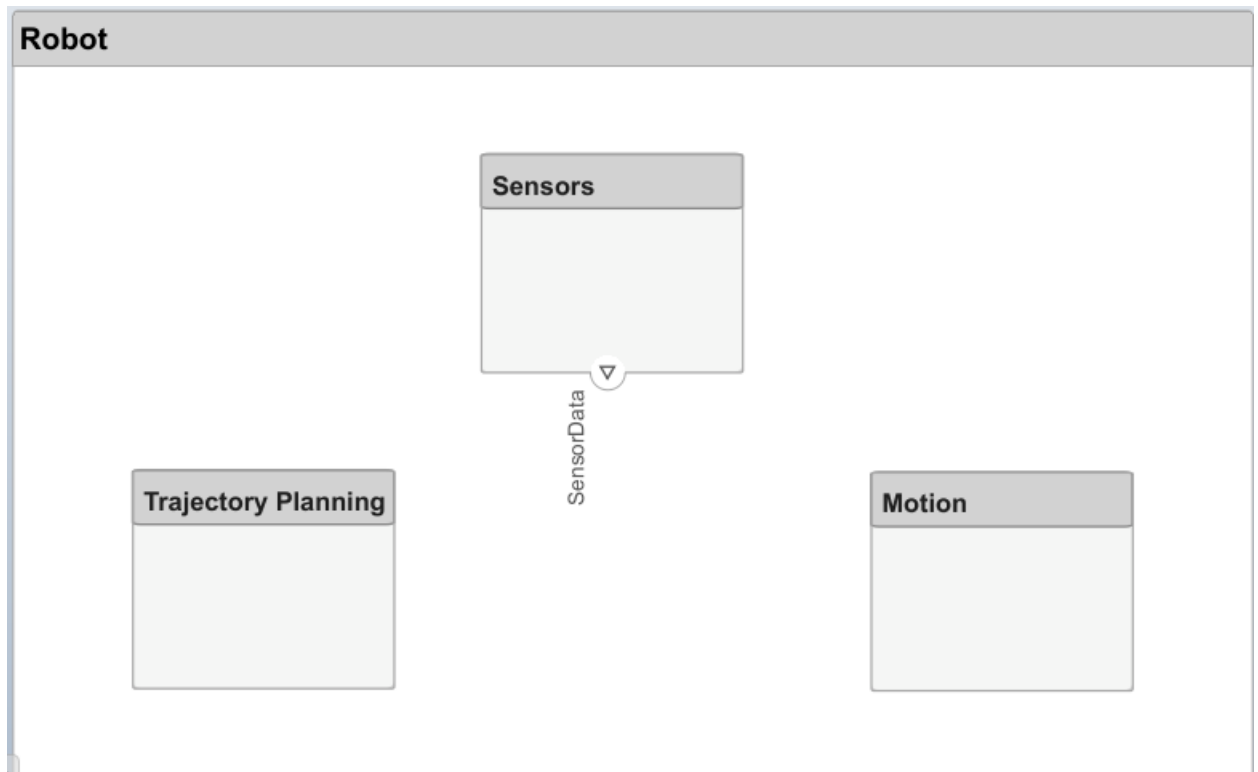
### Create Ports and Connections


You can begin to create connectivity between components by describing the flow of power, energy, data, or any other representative information. Create ports on the components that provide or consume information and connectors that bind two component ports to represent the flow of the information.

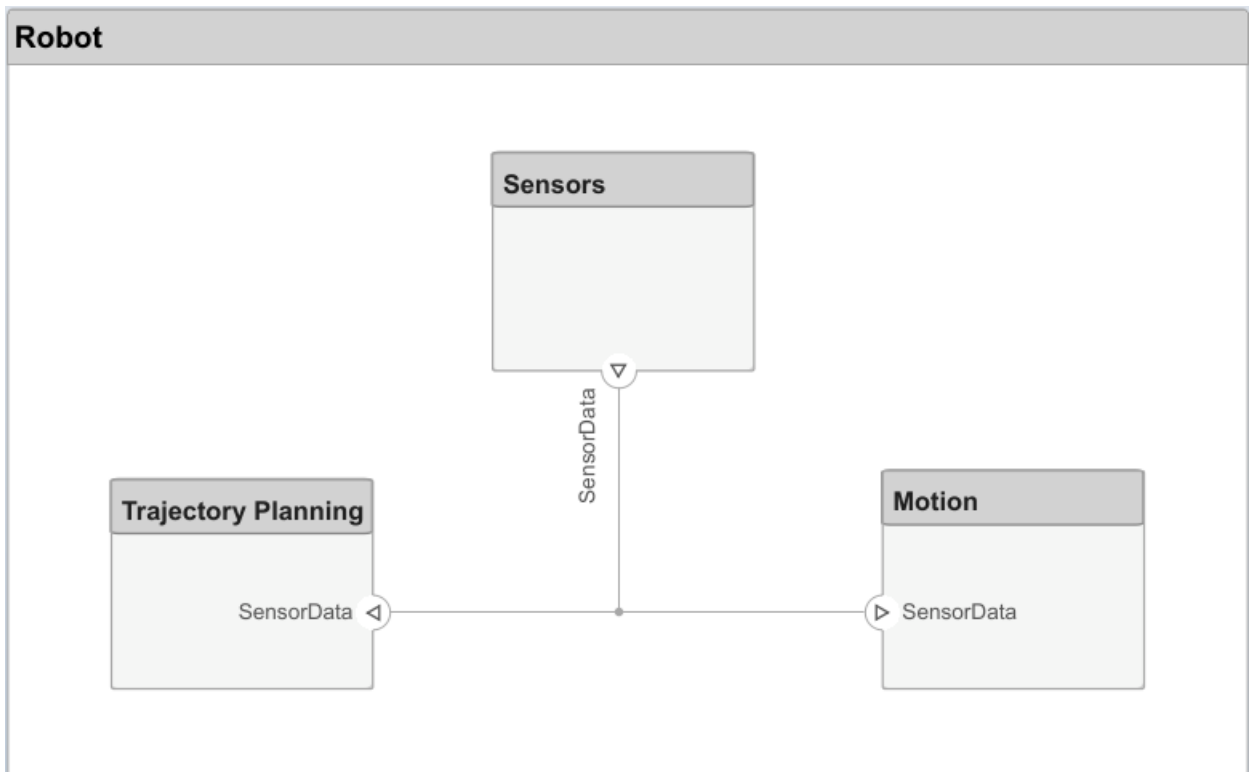
You can add a port to a component on any side, and the port can have either an input or output direction. To create a port, pause your cursor over a component side. Click and release to view port options. Select either **Input**, **Output**, or **Physical** to create a port. Rename the port using a name that represents the information that flows through that port.

- 1 Create an output port on the bottom side of the Sensors component. Rename it **SensorData**.

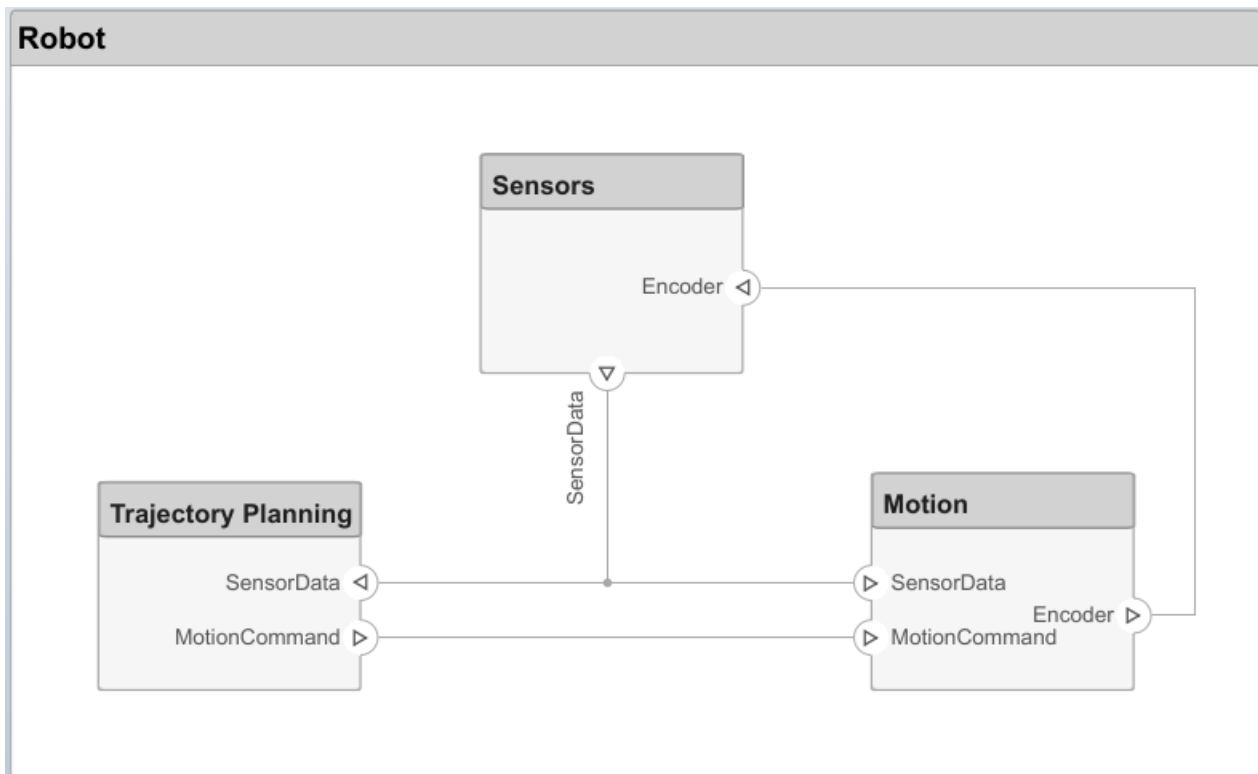




- 2 Click and drag a line from the `SensorData` output port to the `Motion` component. When you see an input port created at the component side, release the mouse button. By default, this new port has the same name as the source port.
- 3 Pause on the corner of the `SensorData` line until you see the branch icon . Right-click and drag a branch line to the `Trajectory Planning` component.



4 Complete the connections as shown in this figure.

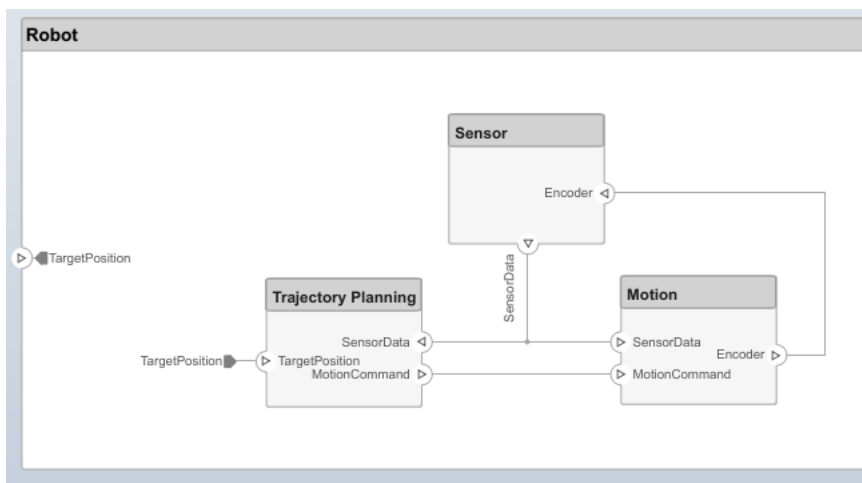


The root level of the architecture model can also have ports that describe the interaction of the system with its environment. In this example, the target position for the robot is provided by a computer external to the robot itself. Represent this relationship with an input port.

- 1 Click the left edge of the architecture model and enter the port name `TargetPosition`.



- 2 Connect an architecture port to a component by dragging a line from the `TargetPosition` input port to the `Trajectory Planning` component. Connections to or from an architecture port appear as tags.



## Edit Data Interfaces

You can define a data interface to fully specify a connection and its associated ports. A data interface can consist of multiple data elements with various dimensions, units, and data types. To check for consistency when connecting a port, you can also associate interfaces with unconnected ports during component design.

Specify the information flow through a port between components by configuring the data interface with attributes. A data interface can be as simple as sending an integer value, but it can also be a set of numbers, an enumeration, a combination of numbers and strings, or a bundle of other predefined interfaces.

Consider the data interface between the `Sensors` and the `Motion` components. The sensor data consists of:

- Position data from two motors
- Obstacle proximity data from two sensors
- A time stamp to capture the freshness of the data

The data has these specifications.

Name	Data Type	Unit
timestamp	double	seconds
position1 for motor 1	double	degrees
position2 for motor 2	double	degrees
distance1 for sensor 1	double	meters
direction1 for sensor 1	double	degrees
distance2 for sensor 2	double	meters
direction2 sensor 2	double	degrees

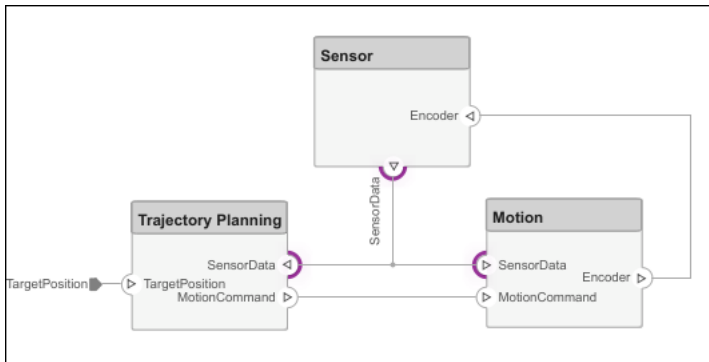
1 Navigate to **Modeling > Design > Interface Editor** to open the Interface Editor.


2 Click the  button to add a data interface. Name the interface `sensordata`.

The data interface is named and defined separately from a component port and then assigned to a port.

3 Click the `SensorData` output port on the `Sensors` component. In the Interface Editor, right-click `sensordata` and select **Assign to Selected Port(s)**.


If you click `sensordata` again, the three `SensorData` ports are highlighted, indicating the ports are associated with that interface.

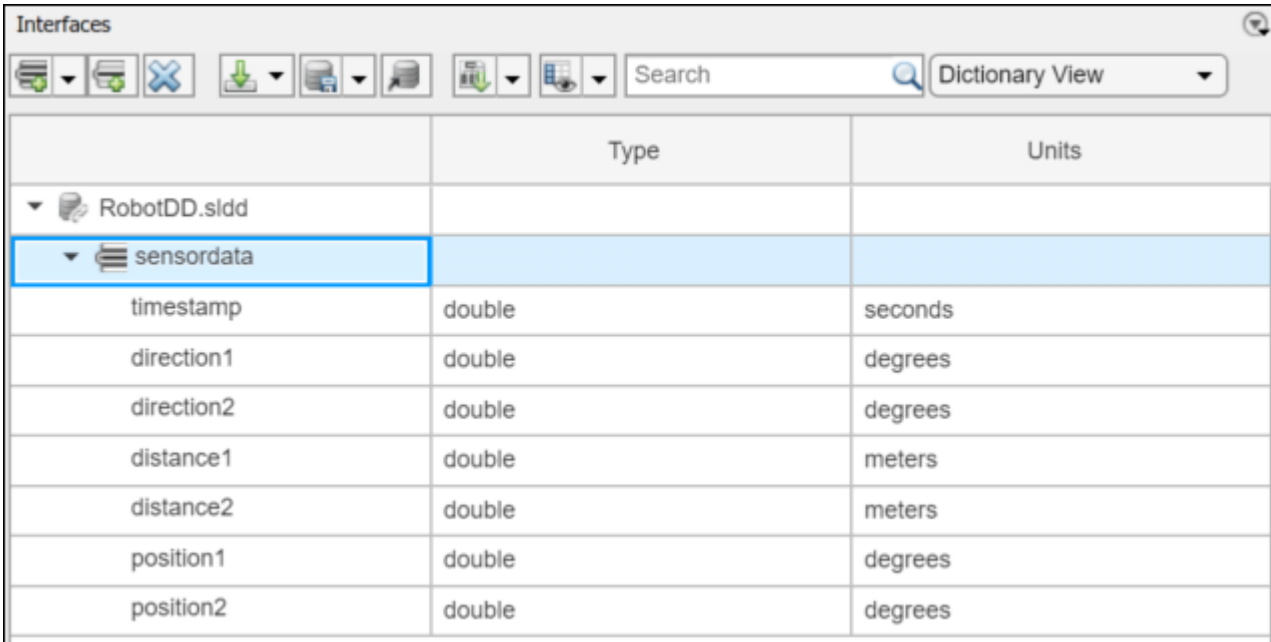


4 Add a data element to the selected data interface. Click the  button to add a data element and name it `timestamp`.

5 Continue adding data elements to the data interface as specified by clicking the add data element button.

6 Edit the properties of a data element in the Interface Editor. Click on the cell corresponding to the data element in the table and add units as shown in the specification.

Click the drop-down next to the  button to save the data interface to a data dictionary. A data dictionary allows you to collectively manage and share a set of interfaces among models. For instance, later in the design, if you choose to model the external computer as a separate architecture model, then this model and the `Robot` model can share the same data dictionary. Here, the dictionary is saved as `RobotDD`.



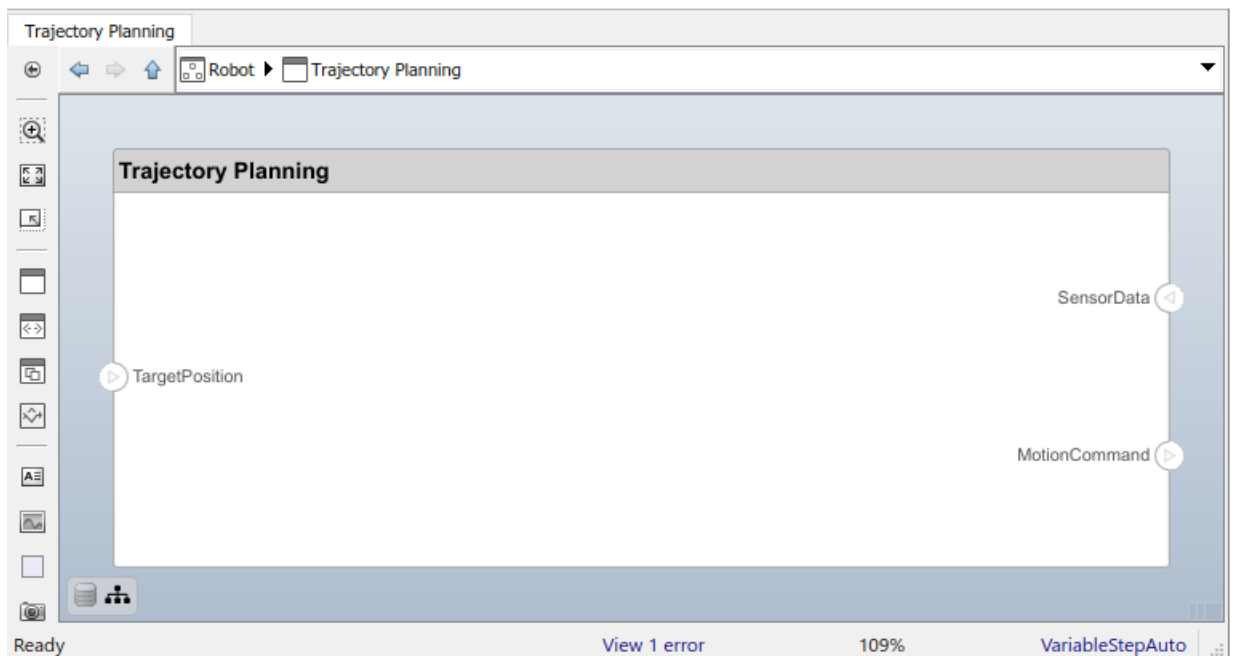
The screenshot shows the 'Interfaces' window with a toolbar and a table. The table lists parameters for the 'sensordata' interface, including their types and units.

	Type	Units
RobotDD.sidd		
sensordata		
timestamp	double	seconds
direction1	double	degrees
direction2	double	degrees
distance1	double	meters
distance2	double	meters
position1	double	degrees
position2	double	degrees

## Decompose Components

Each component can have its own architecture. Double-click a component to decompose it into its subcomponents.

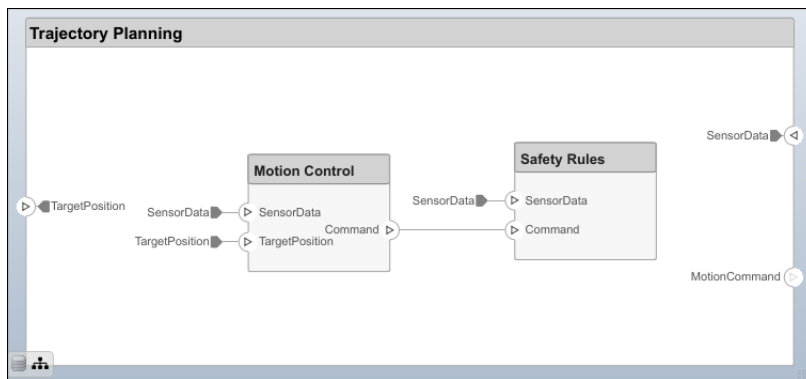
- 1 Double-click the Trajectory Planning component. The title or **Model Browser** indicates the position of the component in the model hierarchy.



This component first uses the motor position data that is part of the `sensordata` interface to compute the ideal position and velocity command. It then processes the obstacle distance information in the same interface to condition this motion command according to some safety rules.

- 2 Add `Motion Control` and `Safety Rules` components as part of the `Trajectory Planning` architecture.

Drag the `TargetPosition` port to the `Motion Control` component. Add a `Command` output port to `Motion Control`, then drag a line to the `Safety Rules` component. Drag lines from the `SensorData` port to the `Motion Control` and `Safety Rules` components.

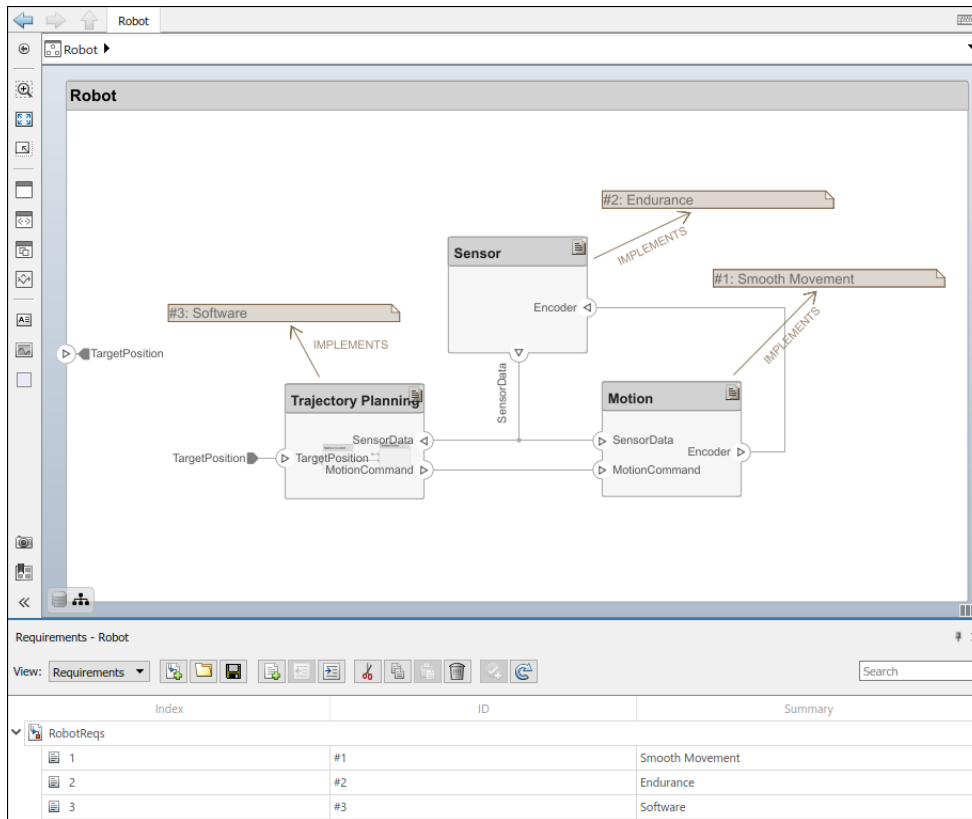


## Manage Requirement Links

Requirements are integral to the systems engineering process. Some requirements relate to the functionality of the overall system, and some relate to aspects of performance such as power, size, and weight. Decomposing high-level requirements into low-level requirements and deriving additional requirements is crucial to defining the architecture of the overall system. For instance, the overall power consumption of the robot determines the requirement for the power consumption of the robot controller.

To allocate and trace requirements with system elements, System Composer fully integrates with Simulink Requirements. To derive appropriate requirements, you must sometimes analyze and specify properties (such as power) for elements of the system including components, ports, or connectors. For example, if the total cost of the system is a concern, a `unitPrice` property is necessary.

Manage requirements from the Requirements perspective in System Composer using Simulink Requirements. Navigate to **Apps > Requirements Manager**.



To enhance the traceability of requirements, link requirements to architectural components and ports. When you click a component in the Requirements perspective, linked requirements are highlighted. Conversely, when you click a requirement, the linked components are shown. To directly create a link, drag a requirement onto a component or a port.

## See Also

## More About

- “System Composer Concepts” on page 3-2

## Extend Architectural Design Using Stereotypes

### In this section...

“Mobile Robot Architecture Model” on page 2-19

“Load Architecture Model Profile” on page 2-19

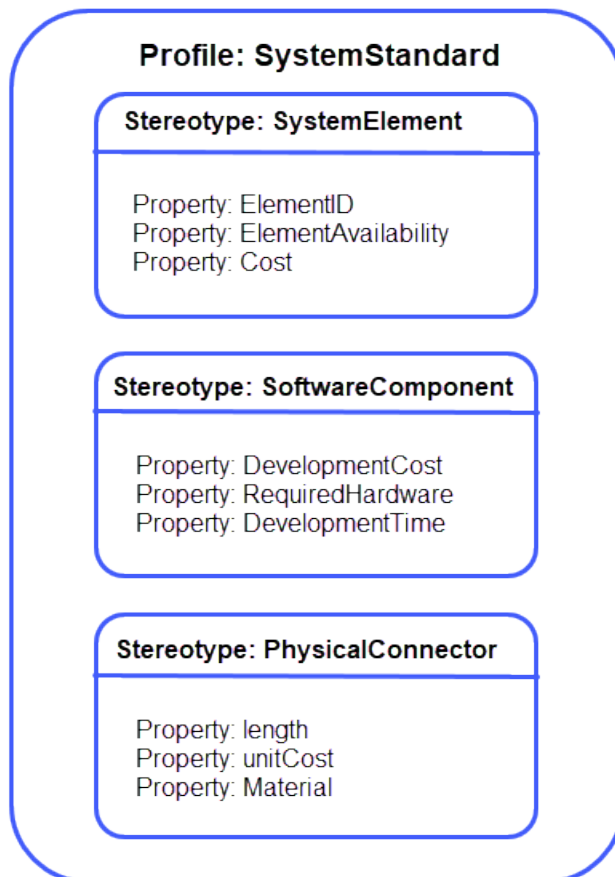
“Apply Stereotypes to Model Elements” on page 2-21

“Set Properties” on page 2-23

“Mobile Robot Architecture Model with Properties” on page 2-25

You can add the `unitPrice` property to an electrical component using a stereotype. A stereotype extends the modeling language with domain-specific metadata. A stereotype adds properties to the root-level architecture, component architecture, ports, connectors, data interfaces, and value types. You can also apply a stereotype to only a specific element type, such as component architectures. When a model element has a stereotype applied, you can specify property values as part of its architectural definition. In addition to allowing you to manage properties relevant to the system specification within the architecture model, stereotypes and associated properties also allow you to analyze an architecture model.

A profile contains a set of model element stereotypes with custom properties.



Each profile contains a set of stereotypes, and each stereotype contains a set of properties.



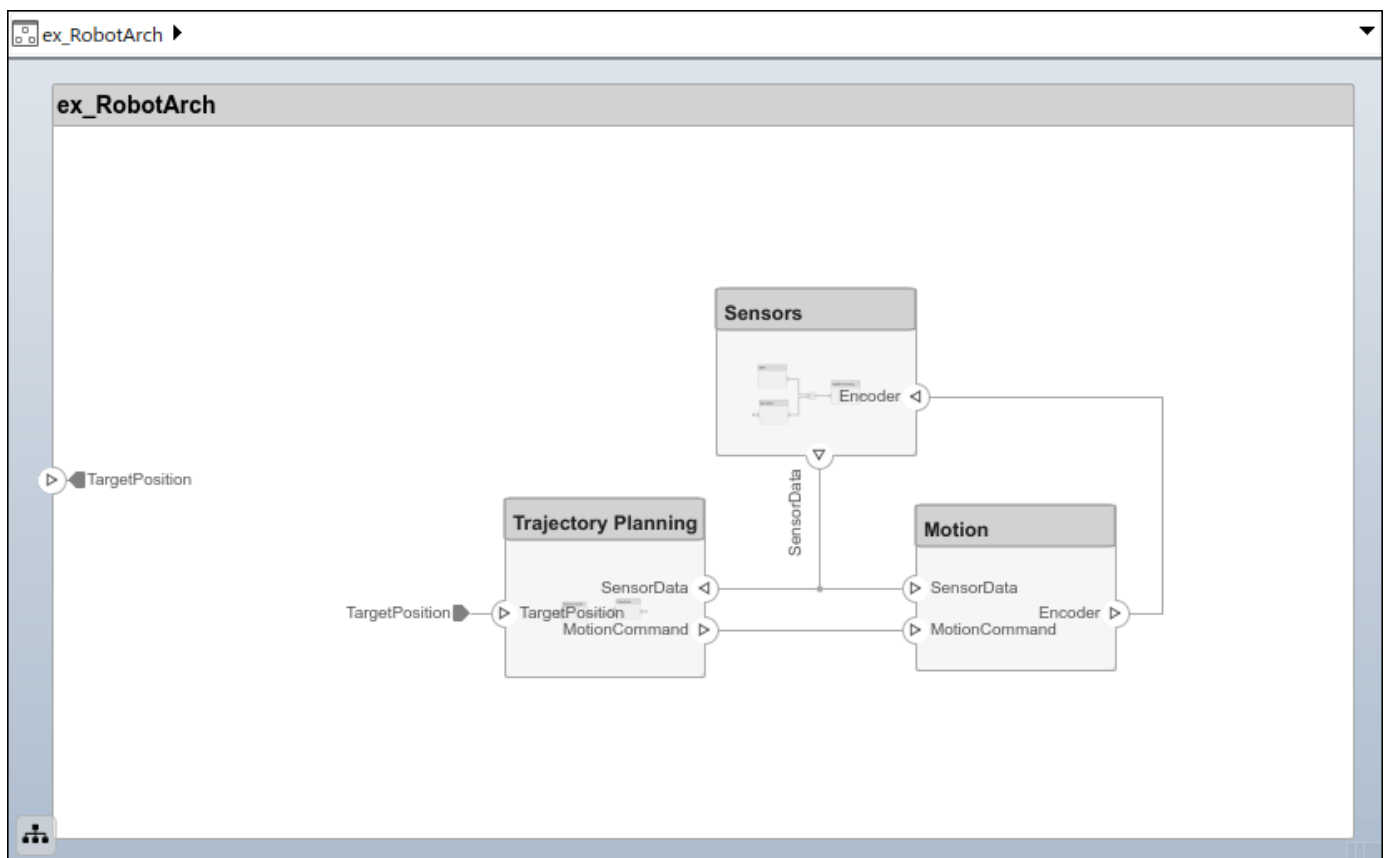
This example will show you how to compute the total cost of the system given the cost of its constituent parts. The example profile is limited to this goal.

Start this tutorial with the following mobile robot architecture model without a profile applied. Use the model to follow the steps and populate its elements with stereotypes and properties.

## Mobile Robot Architecture Model

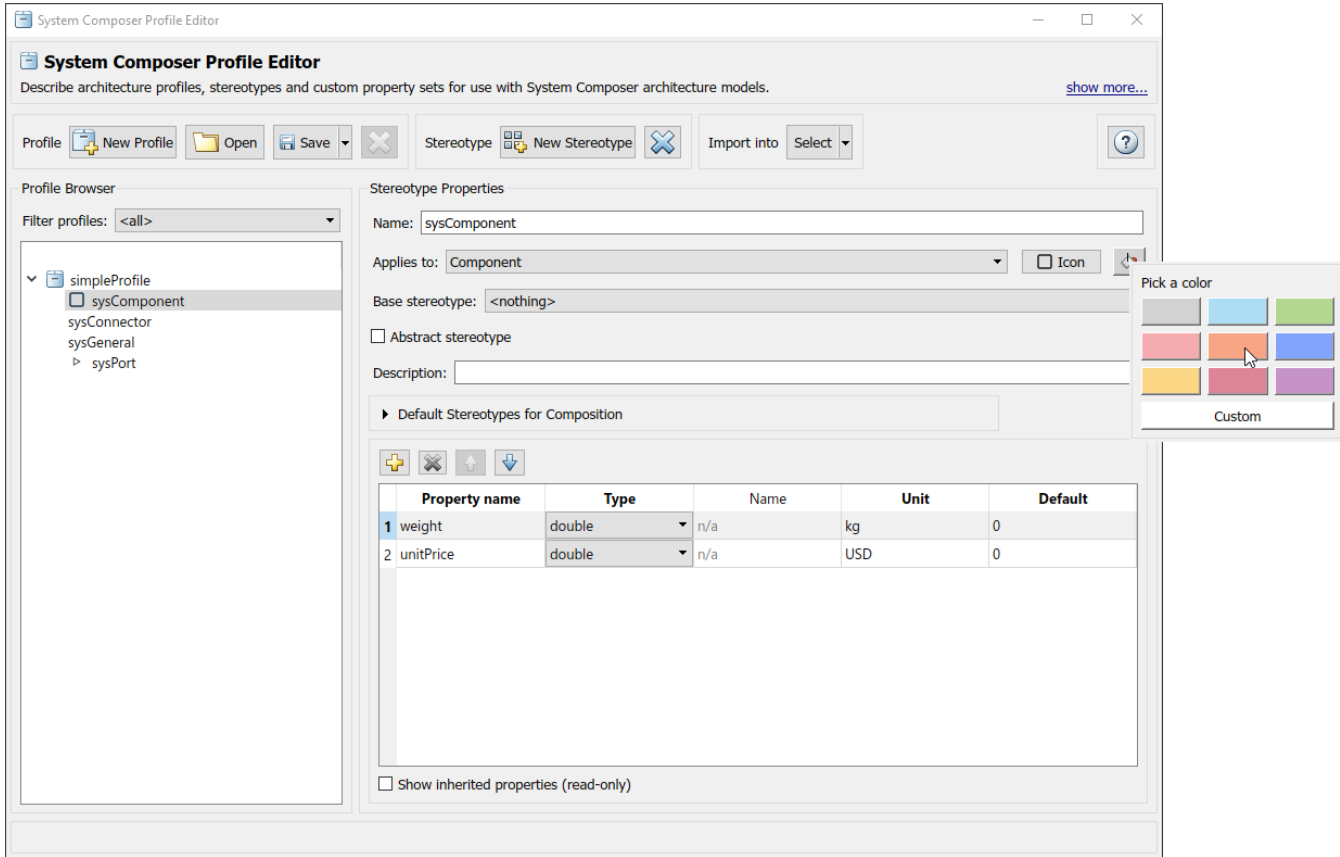
This example shows a mobile robot architecture model with no properties defined. You can apply the stereotypes from the profile `simpleProfile.xml`.

Use the Property Inspector to set the properties on each component.



## Load Architecture Model Profile

Load a profile to make stereotypes available for model elements. This procedure uses the model `ex_RobotArch.slx`. Navigate to **Modeling > Profiles > Profile Editor** to open the Profile Editor. Open the profile file `simpleProfile.xml` to load the profile in the Profile Editor.



In the profile, observe these stereotypes.

Stereotype	Application	Properties
sysGeneral	components, ports, connectors	ID (integer, no units)
		Note (string, no units)
sysComponent	components	weight (double, kg)
		unitPrice (double, USD)
sysConnector	connectors	length (double, m)
		weight (double, kg/m)
		unitPrice (double, USD/m)

Importing the profile makes stereotypes available to their applicable elements.

- `sysGeneral` is a general stereotype, applicable to all element types, that enables adding generic properties such as a `Note`, which project members can use to track any issues with the element.
- `sysComponent` stereotype applies only to components, and includes properties such as `weight` and `cost` that contribute to the total weight and cost specifications of the robot system.
- `sysConnector` stereotype applies to connectors and includes `unitPrice` and `weight` properties defined per meter of length (assuming a physical connector, such as a wire). These properties help compute the total weight and cost of the design.


- sysPort stereotype applies to ports and does not include any properties.

**Note** You can add a stereotype icon to all component-level stereotypes. You can choose from a set of default icons, or you can create your own icons.

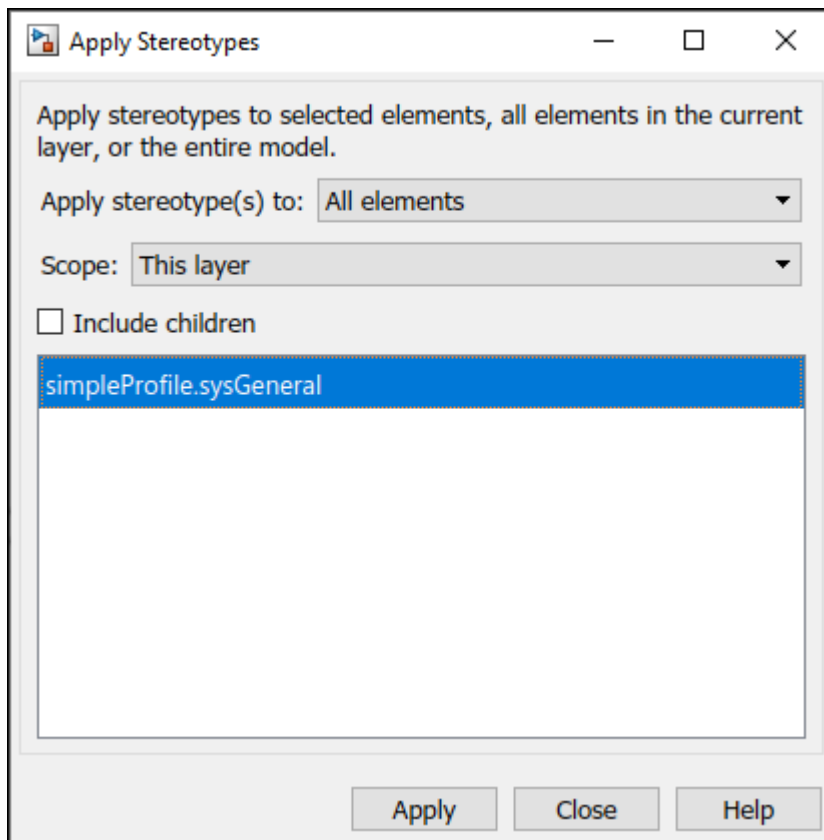


## Apply Stereotypes to Model Elements

Add custom properties to a model element by applying a stereotype from a loaded profile.

- 1 Navigate to **Modeling > Profiles > Import** .
- 2 Select simpleProfile.
- 3 Open the Sensors component.
- 4 Navigate to **Modeling > Profiles > Apply Stereotypes** to open the Apply Stereotypes dialog box.
- 5 In Apply Stereotypes, from **Apply stereotype(s) to**, select All elements. From **Scope**, select This layer.

In the list of available stereotypes, select simpleProfile.sysGeneral.



- 6 Click **Apply** and close the window to exit the dialog box.
- 7 Select the GPS component. Right-click, then click **Apply Stereotype**. Select the `simpleProfile.sysComponent` stereotype.

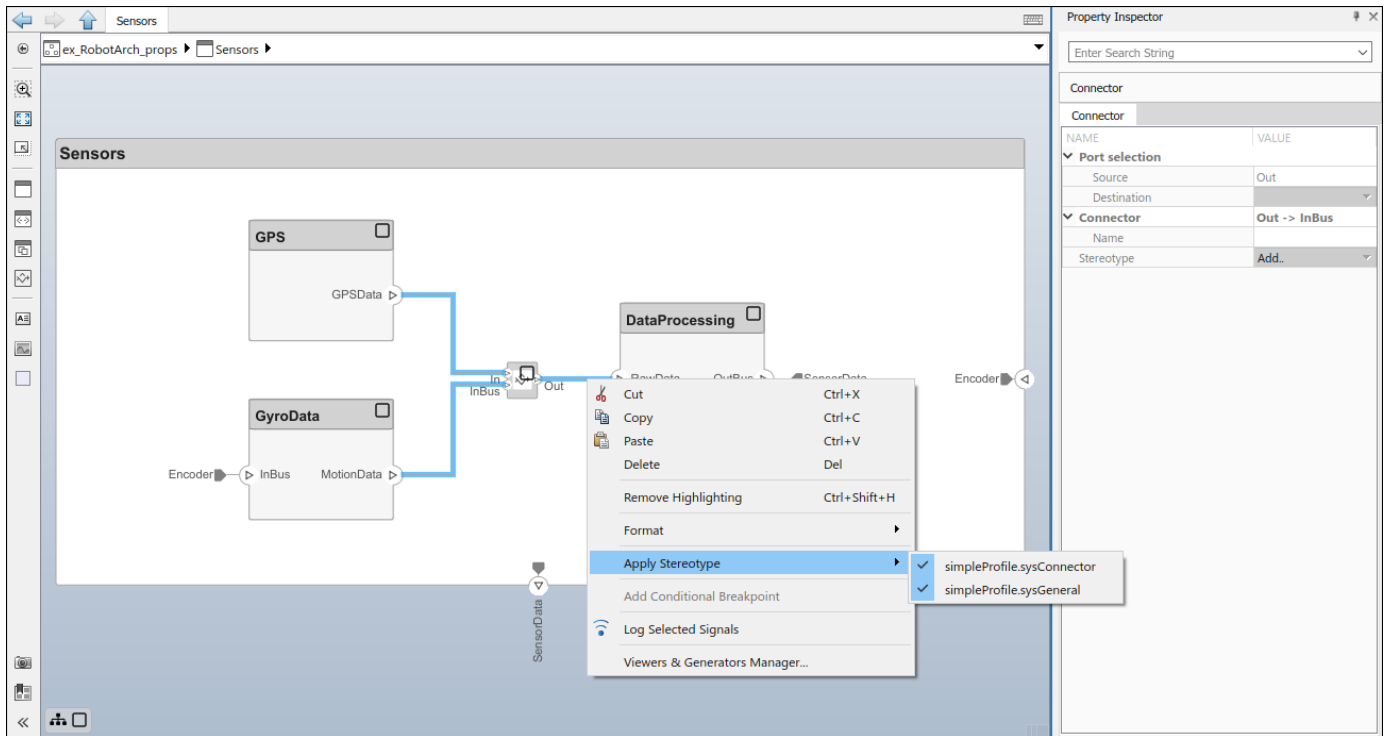
---

**Note** The `sysComponent` stereotype is used for managing physical properties and cost.

---

Repeat these steps for the `GyroData` and `DataProcessing` components.

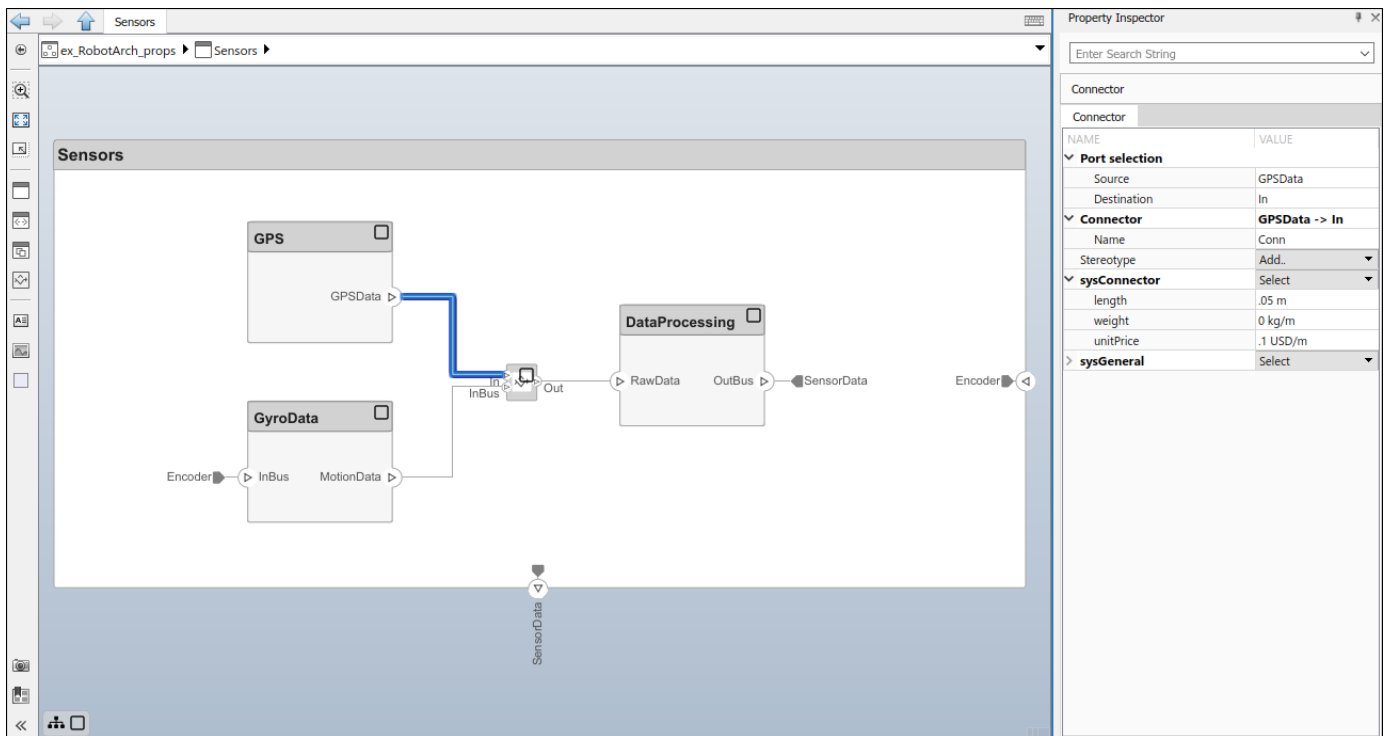
- 8 Navigate to the top of the model. Apply the `sysComponent` stereotype to the `Sensors` and `Trajectory Planning` components and the top-level architecture model. Right-click each component or a space in the top-level, then select **Apply Stereotype** to ensure `simpleProfile.sysComponent` is selected.
- 9 Apply the `sysConnector` stereotype to all connectors in the `Sensors` layer, the `Trajectory Planning` layer, and the top model layer. Press and hold **Shift** to select multiple connectors. Right-click the selection, click **Apply Stereotype**, and select the `simpleProfile.sysConnector` stereotype.



## Set Properties

Set the property values to enable cost analysis. Follow this example for the GPS module.

- 1 In the Sensors component, select the GPS component.
- 2 Open the Property Inspector by navigating to **Modeling > Design > Property Inspector**.
- 3 Expand the sysComponent stereotype to see the properties.
- 4 Set unitPrice to 5 and press **Enter**.
- 5 Select the GPSData port connector. Check that length is set to 0.05 and that unitPrice is set to 0.1.



- 6 Complete the model using the values in this table. If a property is not in the table, it has no effect on the analysis, so you can leave it blank. Pin the Property Inspector to the editor to keep the Property Inspector visible during this operation.

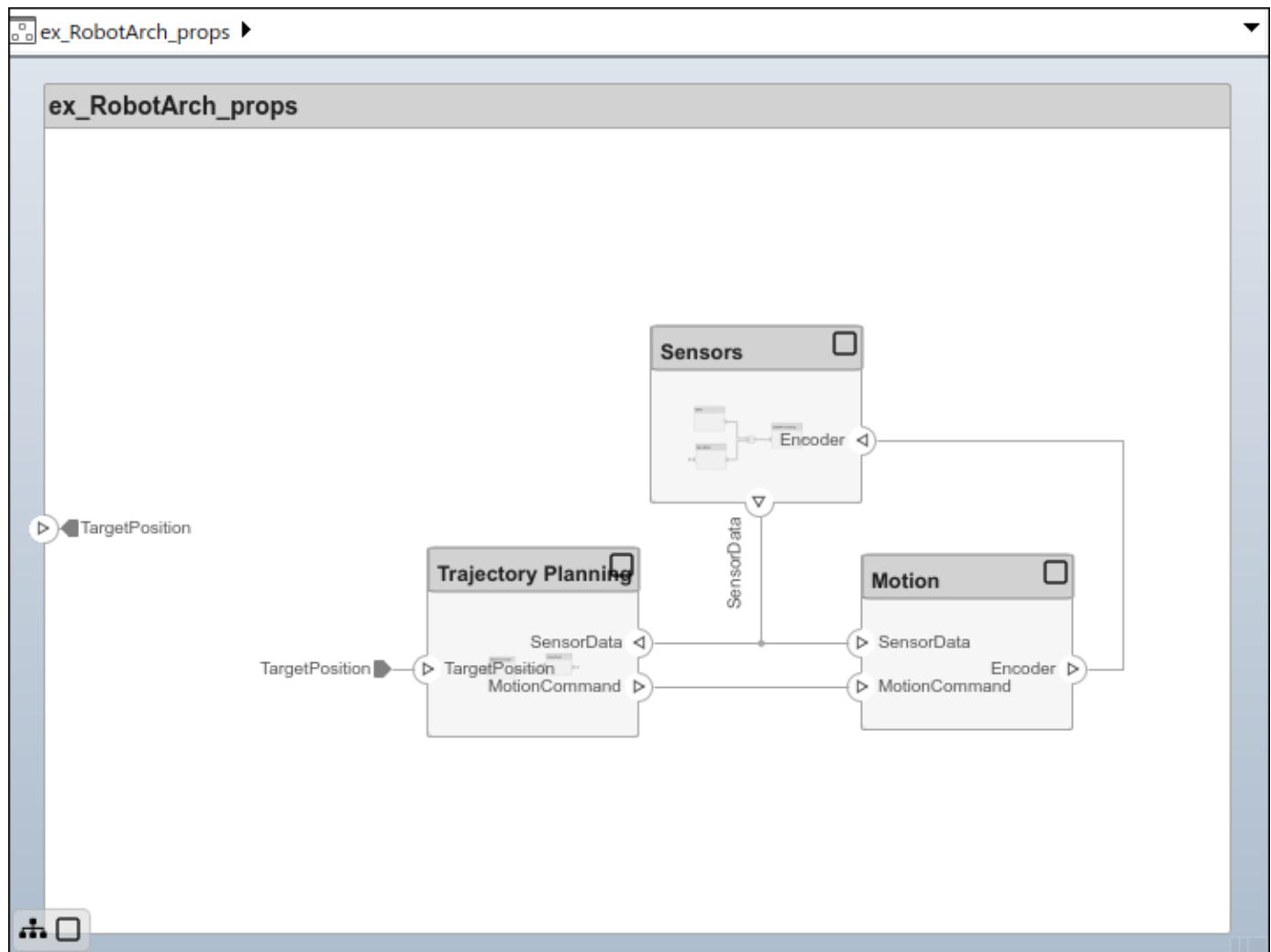
Layer	Element	Property	Value
Top layer	Encoder connector	length	0.5
		unitPrice	0.1
	SensorData connector	length	0.6
		unitPrice	0.2
	MotionCommand connector	length	0.5
		unitPrice	0.2
Sensors component	unitPrice	5	
Trajectory Planning component	unitPrice	500	
Motion component	unitPrice	750	
Sensors layer	GyroData component	unitPrice	50
	DataProcessing component	unitPrice	500
	GPS component	unitPrice	100
	GPSData connector	length	0.05
		unitPrice	0.1
MotionData connector	length	0.05	

Layer	Element	Property	Value
		unitPrice	0.1
	RawData connector	length	0.05
		unitPrice	0.1

The model below reflects the final result of this tutorial. Use this finalized model to perform an analysis and create custom views.

## Mobile Robot Architecture Model with Properties

This example shows a mobile robot architecture model with stereotypes applied to components and properties defined.



## See Also

### More About

- “System Composer Concepts” on page 3-2



## Analyze an Architecture Model with an Analysis Function

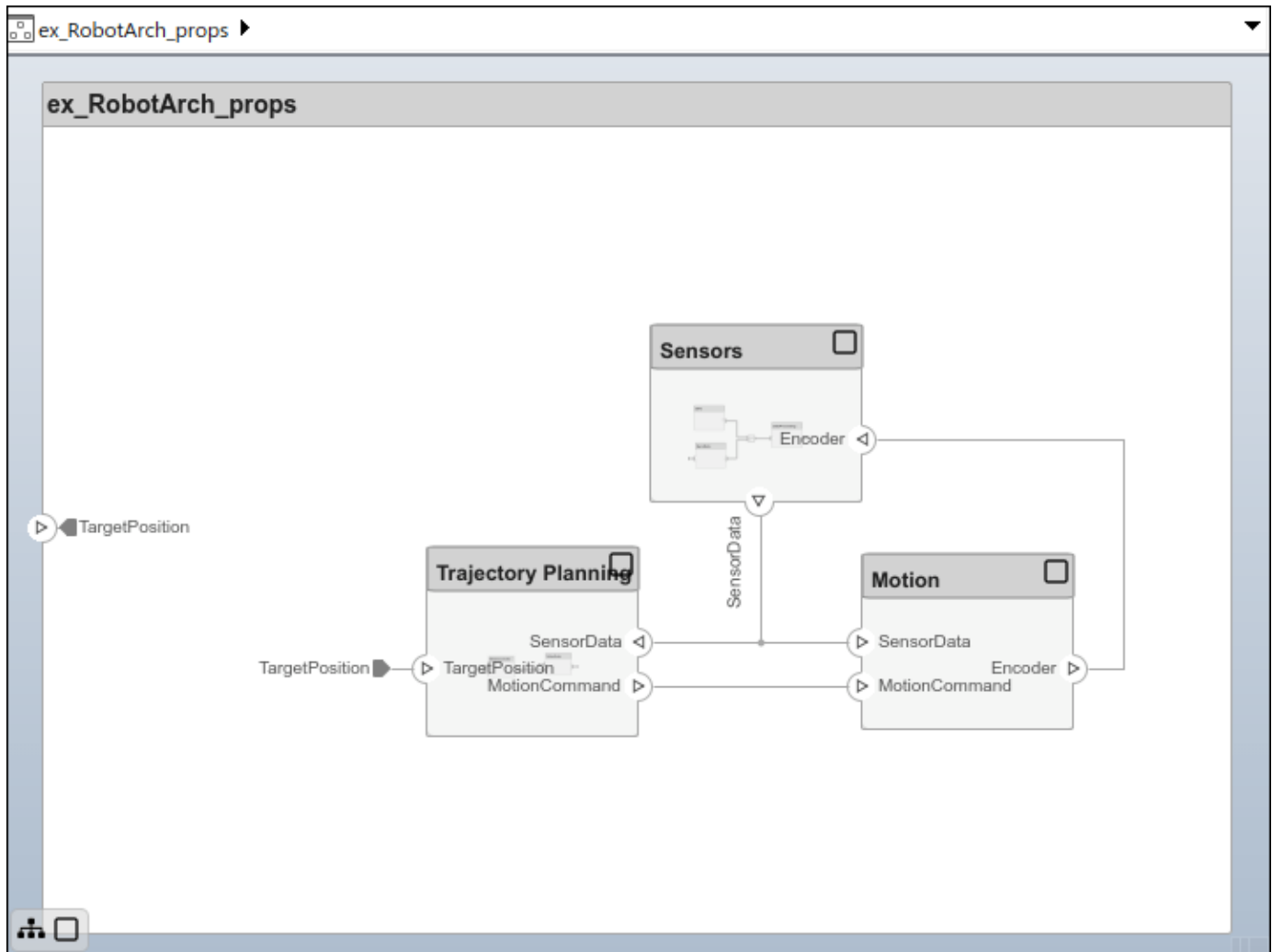
In this section...
“Mobile Robot Architecture Model with Properties” on page 2-27
“Perform an Analysis” on page 2-28

With properties specified on model elements, you can use MATLAB to perform analysis and calculate total cost for all elements within the design. You can then create additional derived requirements for the designers of individual components in the system, such as Trajectory Planning or Sensors.

Perform static analyses based on element properties to perform data-driven trade studies and verify system requirements. Consider a robot architecture model where total cost is a consideration. For this tutorial, you will use the mobile robot architecture model with properties to perform static analysis.


### Mobile Robot Architecture Model with Properties

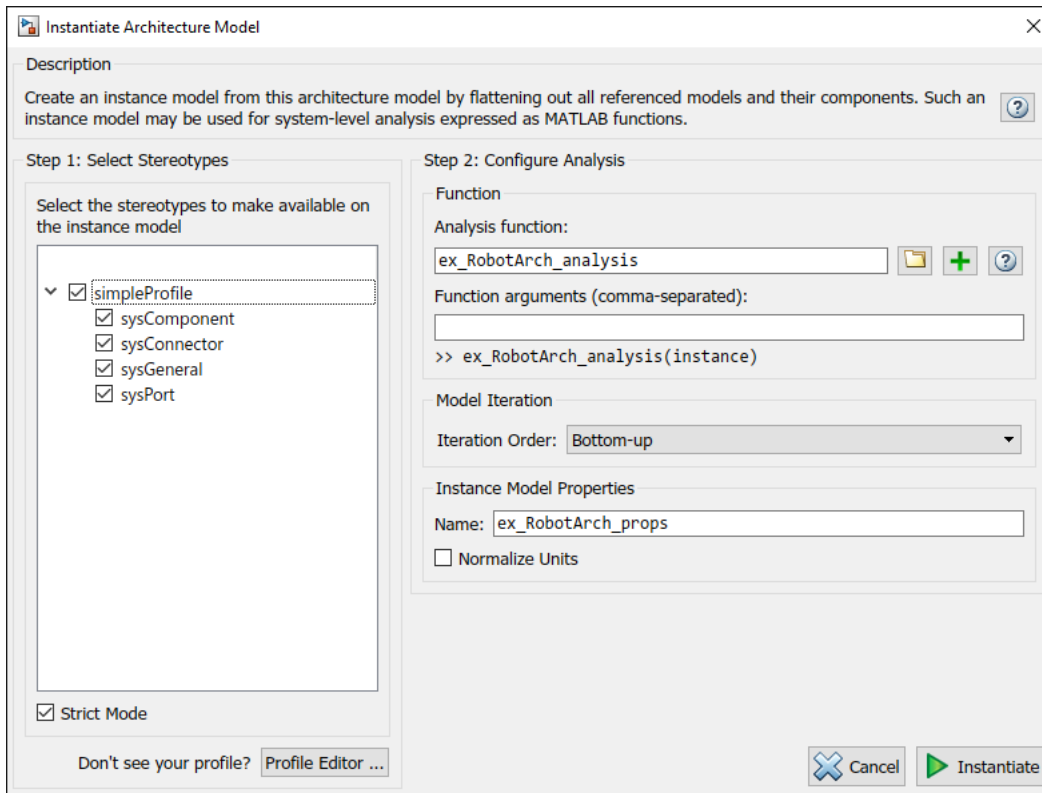
This example shows a mobile robot architecture model with stereotypes applied to components and properties defined.



## Perform an Analysis

Analyze the total cost for all components in the robot model. This procedure uses the model `ex_RobotArch_props.slx`.

- 1 Navigate to **Modeling > Views > Analysis Model** to open the Instantiate Architecture Model dialog.
- 2 Add an analysis function. In the **Analysis function** box, enter the function name `ex_RobotArch_analysis` without an extension, and then click the  button. A MATLAB function file is created and saved with the name `ex_RobotArch_analysis.m`.



The analysis function includes constructs that get properties from model elements, given as a template. Modify this template to add the cost of individual elements and obtain total cost for their parent architecture. This function computes the cost for one model element as a total of its own cost and the cost of all of its child components. Copy and paste the function below into your analysis function.

```
function ex_RobotArch_analysis(instance,varargin)
if instance.isComponent()
    if instance.hasValue("sysComponent.unitPrice") % Check if price is defined
        sysComponent_unitPrice = instance.getValue("sysComponent.unitPrice");
        for child = instance.Components % Iterate through all components
            if child.hasValue("sysComponent.unitPrice")
                comp_price = child.getValue("sysComponent.unitPrice");
                sysComponent_unitPrice = sysComponent_unitPrice + comp_price;
            end
        end
        for child = instance.Connectors % Connectors are priced by length
            if child.hasValue("sysConnector.unitPrice")
                unitPrice = child.getValue("sysConnector.unitPrice");
                length = child.getValue("sysConnector.length");
                sysComponent_unitPrice = unitPrice*length + sysComponent_unitPrice;
            end
        end
        instance.setValue("sysComponent.unitPrice",sysComponent_unitPrice) % Set totals
    end
end
```

- 3 Return to the Instantiate Architecture Model screen and click **Instantiate**. The Analysis Viewer opens and shows the properties of each model element. The default values for the start of the analysis are taken from the property values you entered when you attached the stereotype to the model and edited their values.
- 4 In the **Analysis** section, select **BottomUp** as the iteration method, then click **Analyze**.

The cost of each element is added bottom-up to find the cost of the system. The result is written to the analysis instance and is visible in the Analysis Viewer.

The screenshot shows the Analysis Viewer interface. The main window displays a table with columns: unitPrice, weight, length, unitPrice, weight, and ID. The top row, representing the 'ex\_RobotArch\_props' architecture, has a yellow background and contains the values 1905.285, 0, 0, 0, 0, and 0. Below it, various sub-components like Motion, Sensors, Adapter, DataProcessing, GPS, GyroData, Trajectory Planning, and SafetyRules are listed with their respective values. The right sidebar shows the 'INSTANCE PROPERTIES' for 'ex\_RobotArch\_props', listing properties like 'unitPrice' (1,905.28499999999 USD) and 'weight' (0 kg).

Instance	unitPrice	weight	length	unitPrice	weight	ID
ex_RobotArch_props	1905.285	0	0	0	0	
Motion	750	0	0	0	0	
Sensors	655.015	0	0	0	0	
Adapter	0	0	0	0	0	
DataProcessing	500	0	0	0	0	
GPS	100	0	0	0	0	
GyroData	50	0	0	0	0	
Adapter:Out->DataProcessing:RawData			0.05	0.1	0	
DataProcessing:OutBus->Sensors:SensorData			0	0	0	
GPS:GPSData->Adapter:In			0.05	0.1	0	
GyroData:MotionData->Adapter:InBus			0.05	0.1	0	
Sensors:Encoder->GyroData:InBus			0	0	0	
Trajectory Planning	500	0	0	0	0	
MotionController	0	0	0	0	0	
SafetyRules	0	0	0	0	0	
MotionController:command->SafetyRules:command			0	0	0	
SafetyRules:OutBus->Trajectory Planning:MotionCommand			0	0	0	
Trajectory Planning:SensorData->MotionController:SensorData			0	0	0	
Trajectory Planning:SensorData->SafetyRules:SensorData			0	0	0	
Trajectory Planning:TargetPosition->MotionController:TargetPosition			0	0	0	
Motion:Encoder->Sensors:Encoder			0.5	0.1	0	
Sensors:SensorData->Motion:SensorData			0.6	0.2	0	
Sensors:SensorData->Trajectory Planning:SensorData			0	0	0	
Trajectory Planning:MotionCommand->Motion:MotionCommand			0.5	0.2	0	
ex_RobotArch_props:TargetPosition->Trajectory Planning:TargetPosition			0	0	0	

The total cost is highlighted in yellow as a computed value on the top row representing the ex\_RobotArch\_props architecture.

### See Also

### More About

- “System Composer Concepts” on page 3-2

## Inspect Components in Custom Architecture Views

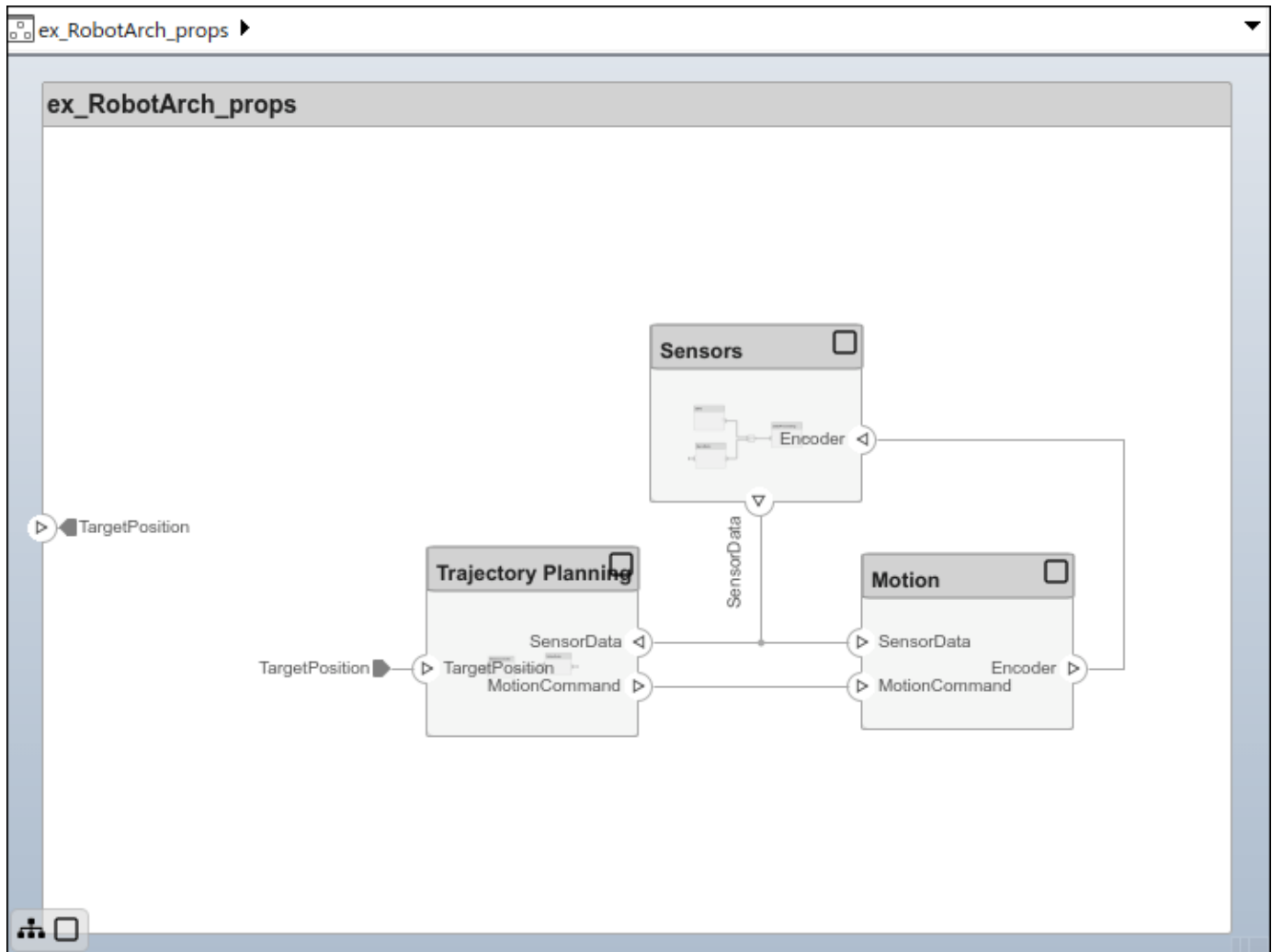
In this section...
“Mobile Robot Architecture Model with Properties” on page 2-31
“Inspect the Component and Its Connectivity” on page 2-32
“Create a Filtered Architecture View” on page 2-34

View the hierarchy and connectivity of a component in a specialized view. Specialized views allow you to create simpler diagrams that show only a subset of the original model elements for a specific design activity or concern.

You will use the System Composer architecture model below in this tutorial.

### Mobile Robot Architecture Model with Properties

This example shows a mobile robot architecture model with stereotypes applied to components and properties defined.



## Inspect the Component and Its Connectivity

Create views dynamically using spotlight views.

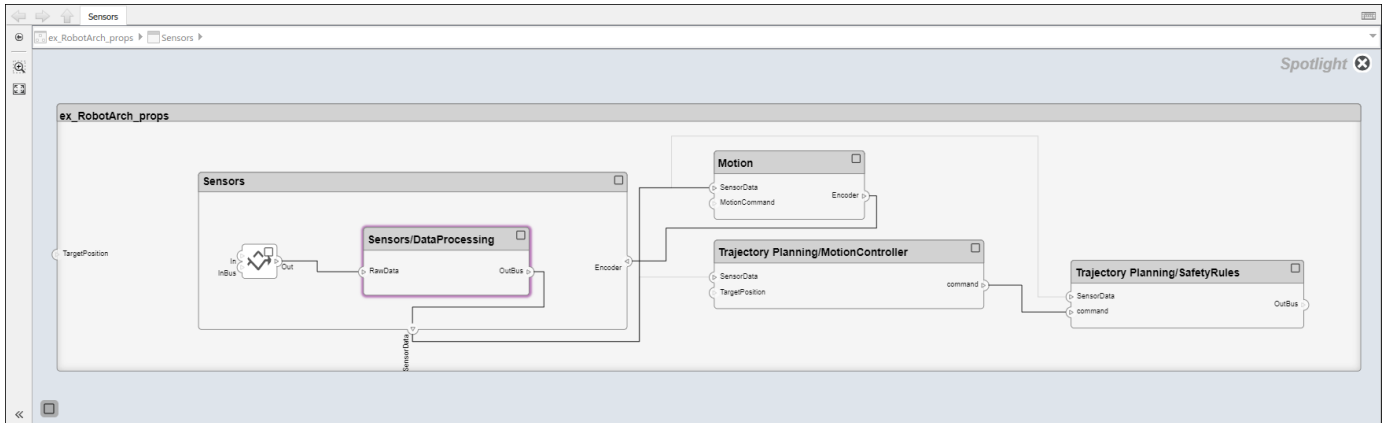
- 1 Double-click the Sensors component, then select the DataProcessing component.
- 2 Right-click the DataProcessing component and select Create Spotlight from Component. Alternatively, select the DataProcessing component and navigate to **Modeling > Views > Architecture Views > Spotlight**.


The spotlight view launches and shows all model elements to which the DataProcessing component connects. The spotlight diagram is laid out automatically and cannot be edited. However, it allows you to inspect just a single component and study its connectivity to other components.

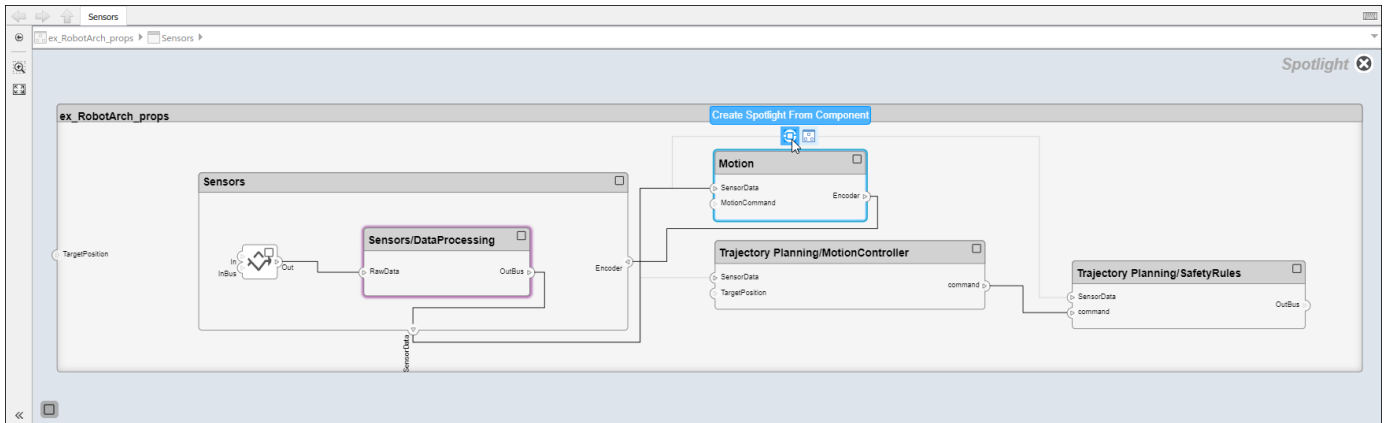
---


**Note** Spotlight views are transient. They are not saved with the model.

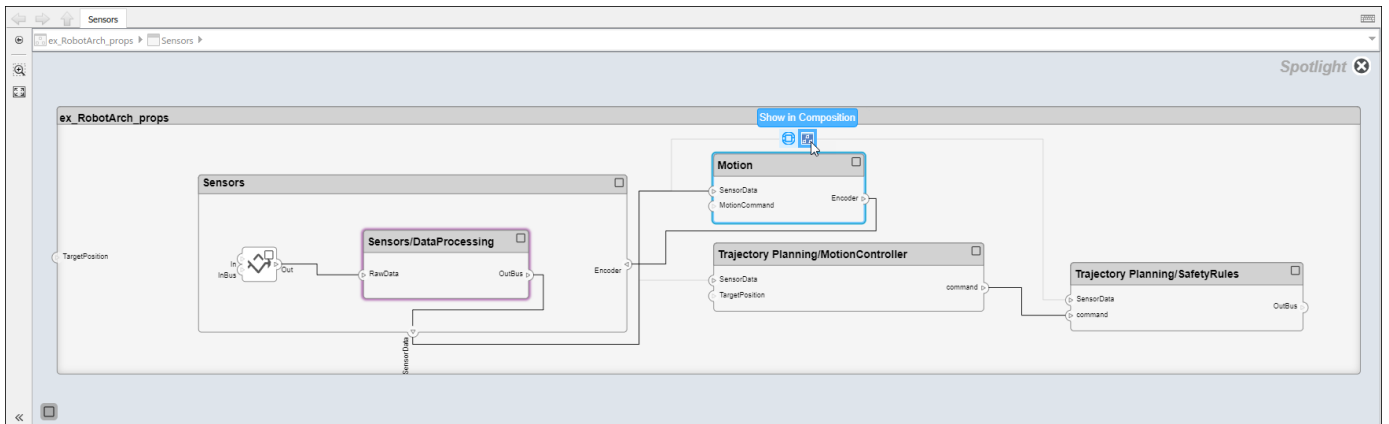
---




3 Shift the spotlight to another component. Select the **Motion** component. Click the ellipsis above the component to open the action menu. To create a spotlight from the component, click .



To view the architecture model at the level of a particular component, select the component and click .

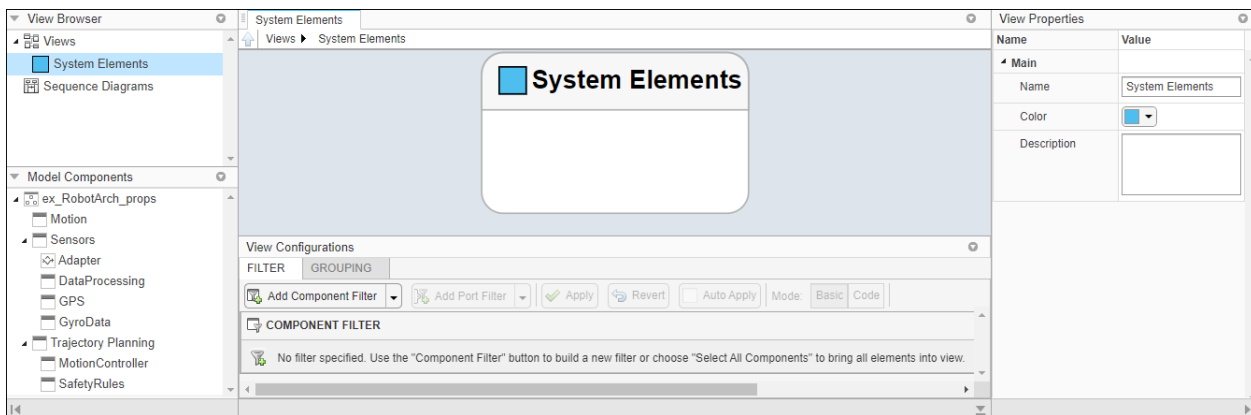


4 Return to the architecture model view by clicking .

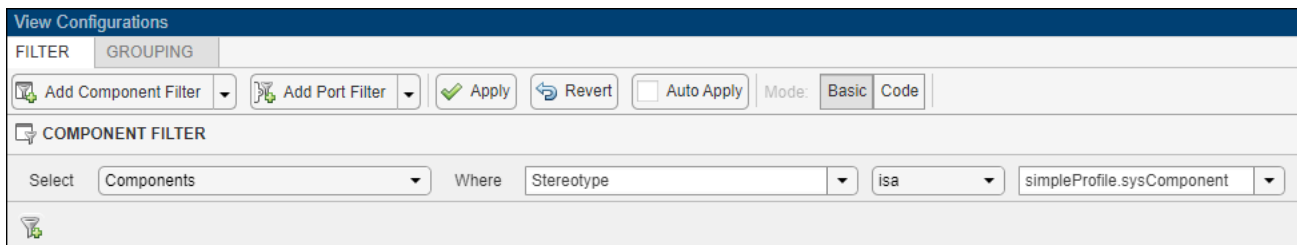
## Create a Filtered Architecture View

Create filtered architecture views to demonstrate specific perspectives with a component diagram or a hierarchy diagram.

- 1 Navigate to **Modeling > Views > Architecture Views** to open the Architecture Views Gallery.
- 2 Select **+** **New > View** to create a new view.
- 3 In **View Properties** on the right pane, in the **Name** box, enter a name for this view, for example, **System Elements**. If necessary, choose a **Color** and enter a **Description**.

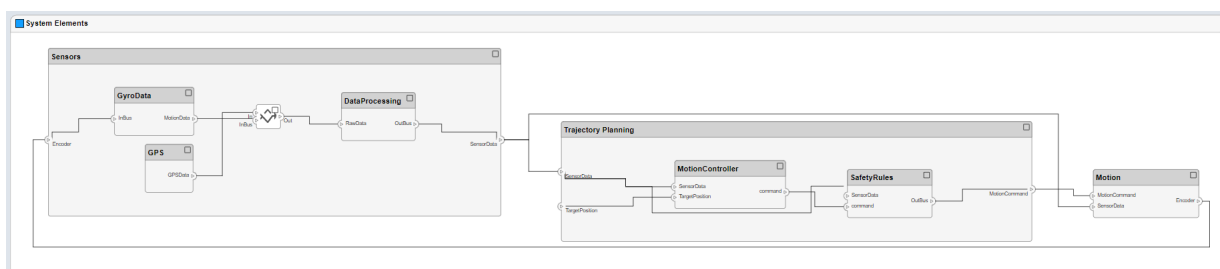


- 4 In the bottom pane, select **View Configurations > Filter > Add Component Filter** to add new form-based criterion to a component filter.
- 5 From the **Select** list, select **Components**. From the **Where** list, select **Stereotype**. In the text box, select **simpleProfile.sysComponent** from the list.



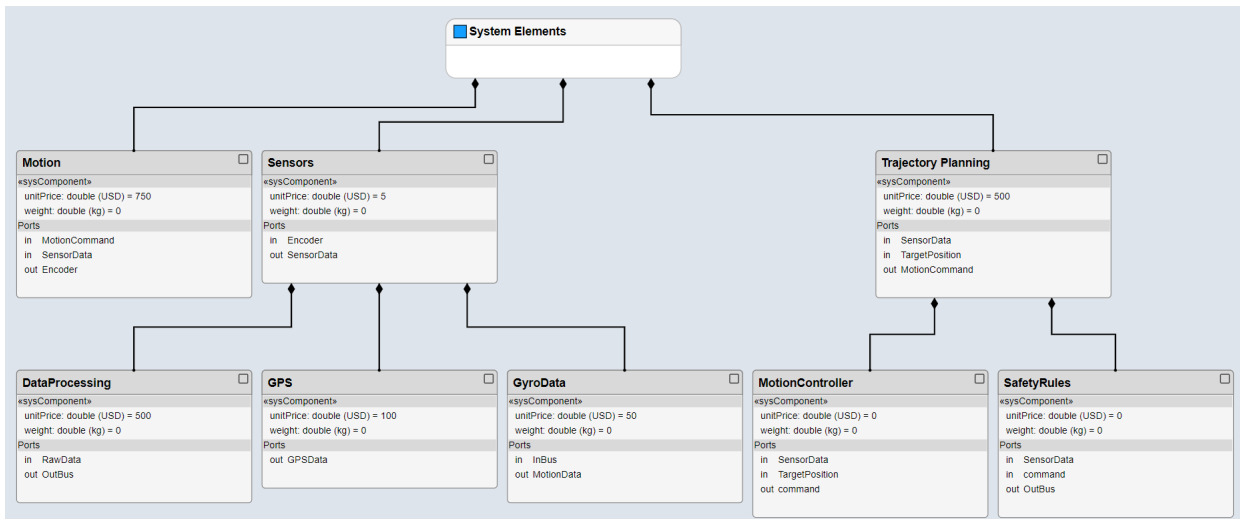
- 6 Click **Apply**

An architecture view is created using the query in the **Component Filter** box. The view is filtered to select all components with the `simpleProfile.sysComponent` stereotype applied to them.

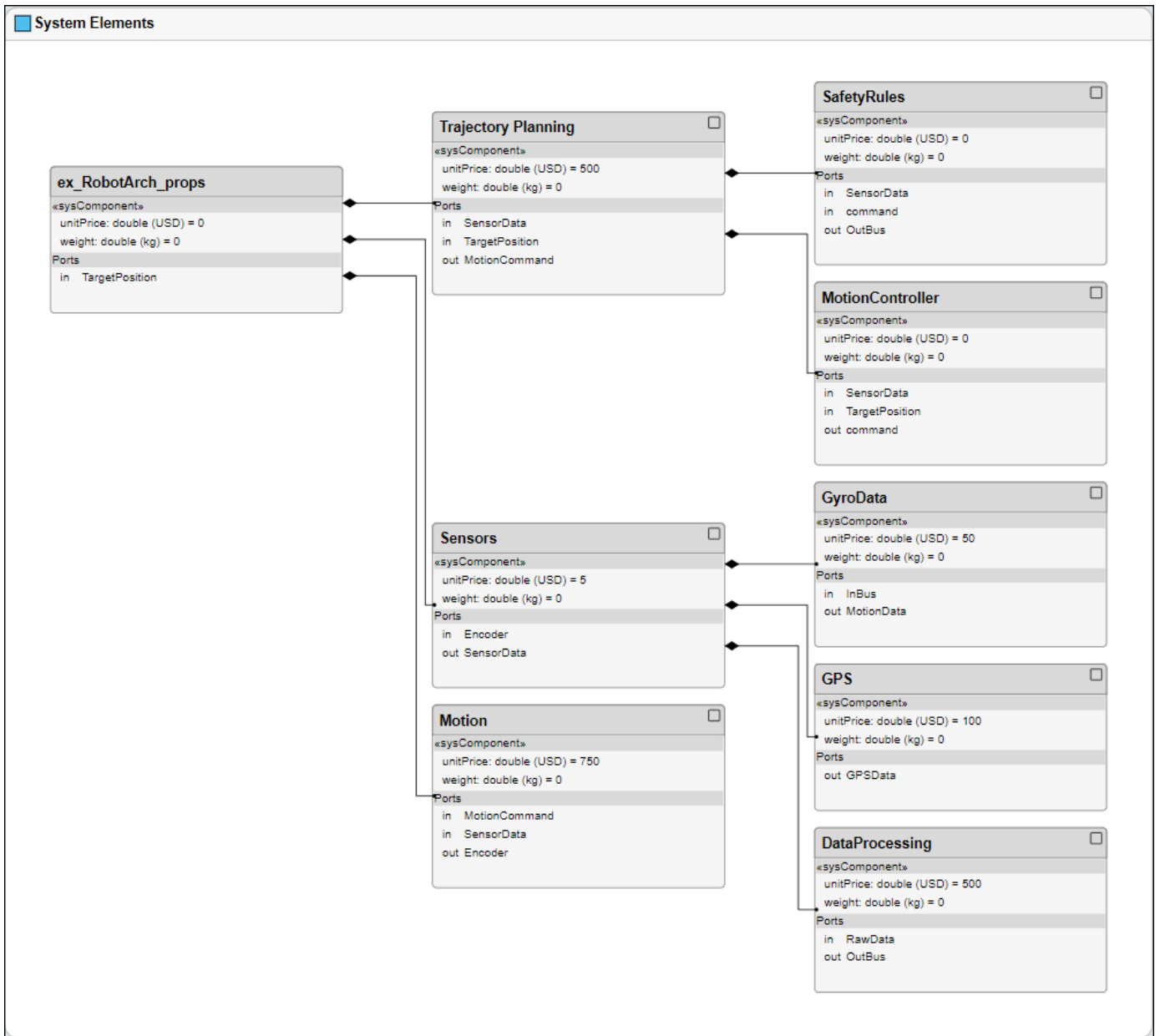




- 7 Navigate to **Diagram > Component Hierarchy** to display the components in tree form with parents above children.



- 8 Navigate to **Diagram > Architecture Hierarchy** to display unique architecture types and their relationships using composition connections.



## See Also

### More About

- “System Composer Concepts” on page 3-2

## Implement Behaviors for Architecture Model Simulation

In this section...
“Robot Arm Architecture Model” on page 2-37
“Reference Simulink Behavior Model in Component” on page 2-38
“Add Stateflow Chart Behavior to Component” on page 2-40
“Design Software Architecture in Component” on page 2-41
“Represent System Interaction Using Sequence Diagrams” on page 2-43

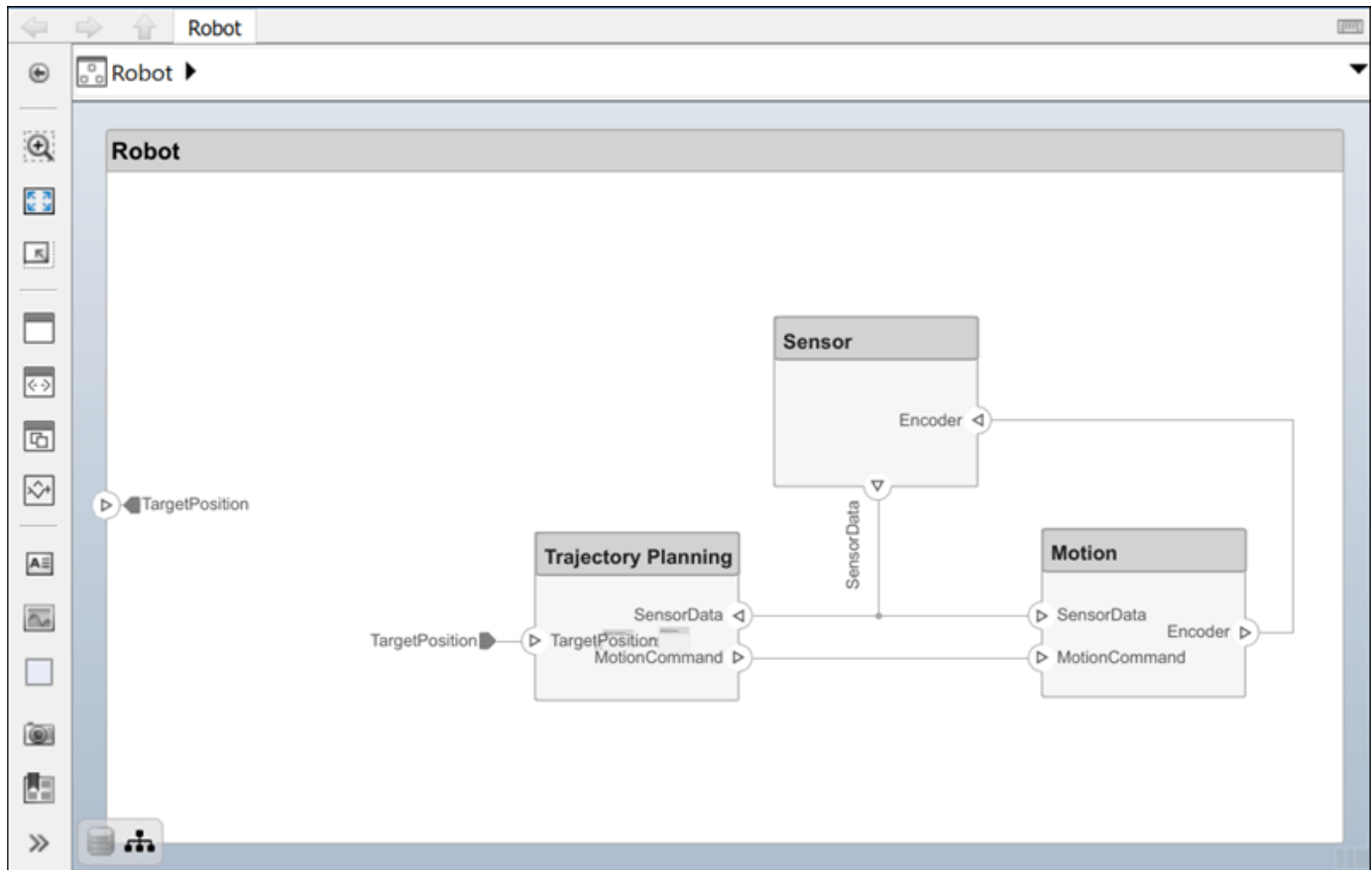
A basic systems engineering workflow in System Composer includes composing an architecture system, defining requirements, adding metadata, performing analyses, and representing the architecture through views. After fulfilling these steps, your system design is closer to meeting stakeholder goals and customer needs.

You can also now begin to design the actual system components using Simulink, Stateflow, and Simscape. You can fully specify, test, and analyze the behavior of a component using the model-based design process.

In this tutorial, you will perform these steps on the robot arm architecture model.

### Robot Arm Architecture Model

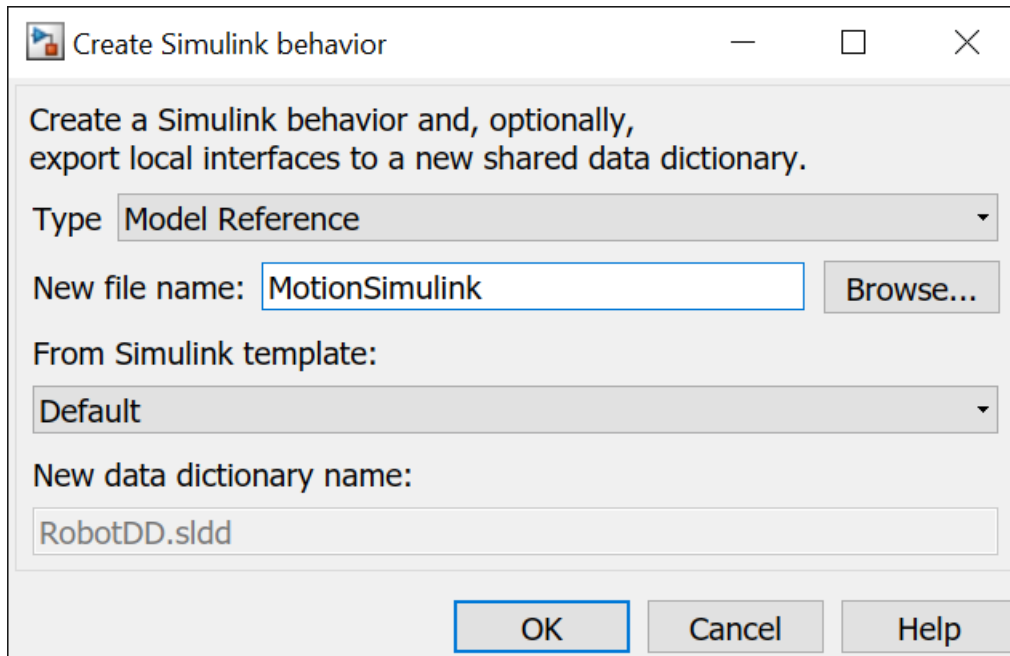
Open the architecture model of a robot arm that consists of sensors, motion actuators, and a planning algorithm. You can use System Composer to view the interfaces and manage the requirements for this model.



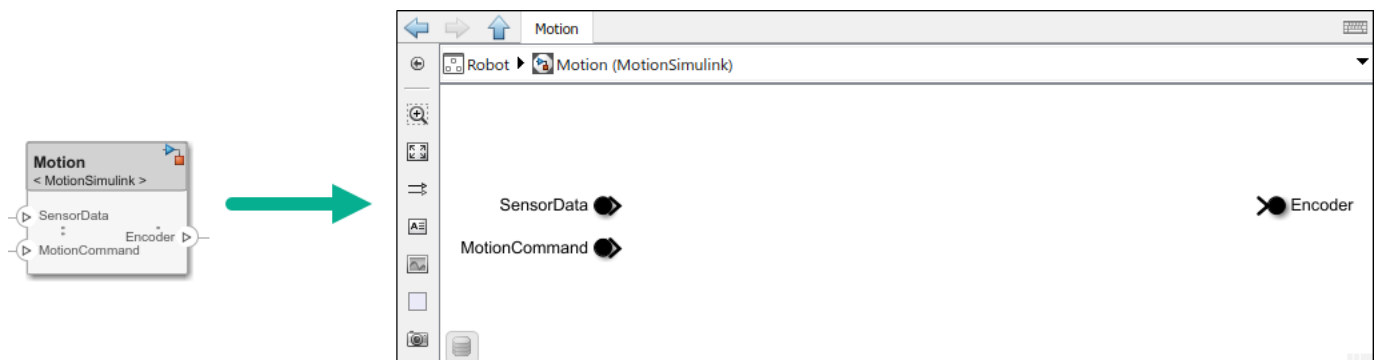
### Reference Simulink Behavior Model in Component

When a component does not require further architectural decomposition, you can enable model simulation and an end-to-end workflow. To enable model simulation, implement Simulink behaviors for components. You can associate a Simulink model with a component or link to an existing Simulink model.

- 1 Right-click the **Motion** component and select **Create Simulink Behavior**. Alternatively, navigate to **Modeling > Component > Create Simulink Behavior**.
- 2 From the **Type** list, select **Model Reference**. Provide the model name **MotionSimulink**. The default name is the name of the component.

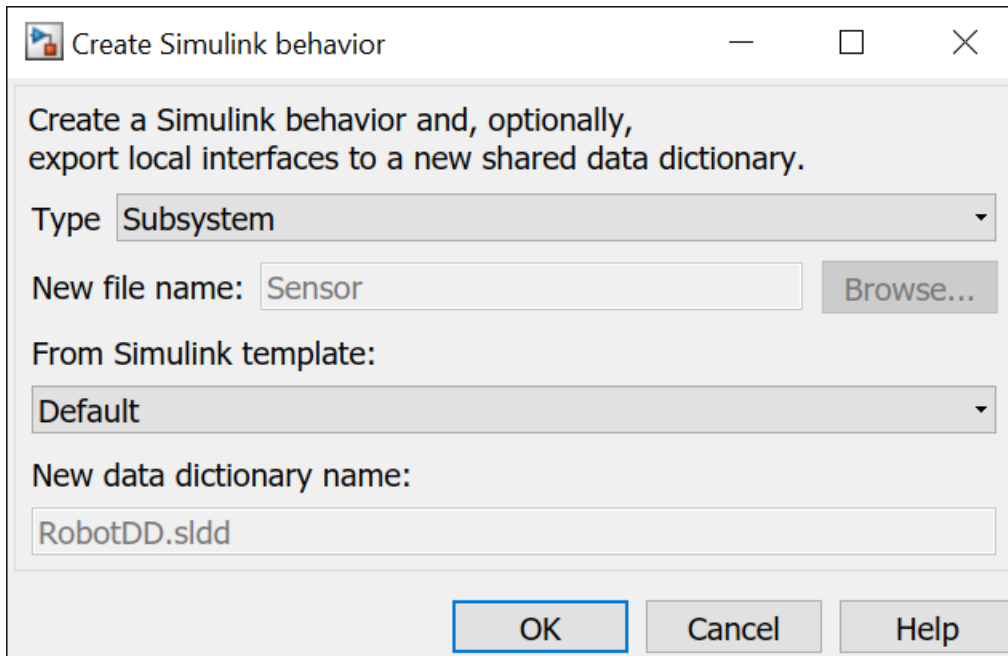


- 3 A new Simulink model with the provided name is created and linked to the component in the architecture model.



You can also 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**. You can type or browse for the name of a Simulink model.

You can also add Simulink subsystem behavior to a component. The new subsystem component is part of the parent System Composer architecture model. From the **Type** list, select **Subsystem**.



Subsystem components are required to author Simscape component behaviors with physical ports, connections, and blocks.

## Add Stateflow Chart Behavior to Component

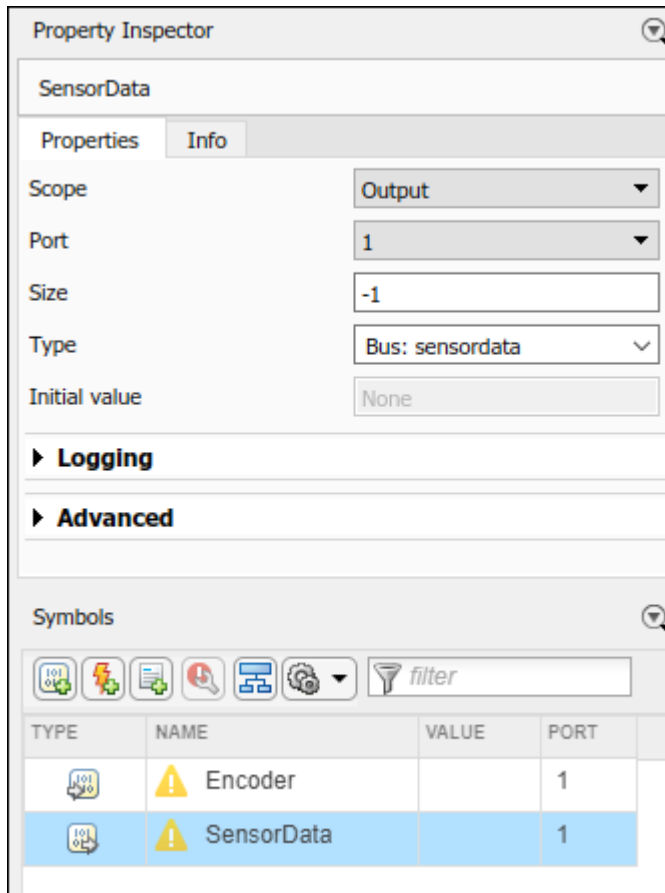
To implement event-based modeling with state machines, add Stateflow chart behavior to a component. State charts consist of a finite set of states with transitions between them to capture the modes of operation for the component. This functionality requires a Stateflow license.

A System Composer component with stereotypes, interfaces, requirement links, and ports, is preserved when you add Stateflow Chart behavior.

- 1 Right-click the **Sensor** component and select **Create Stateflow Chart Behavior**. Alternatively, navigate to **Modeling > Component > Create Stateflow Chart Behavior**.
- 2 Double-click **Sensor**, which has the Stateflow icon. In the **Modeling** menu, select **Design Data**, then click **Symbols Pane** to view the Stateflow symbols. The input port **Encoder** appears as input data in the symbols pane and the output port **SensorData** appears as output data.



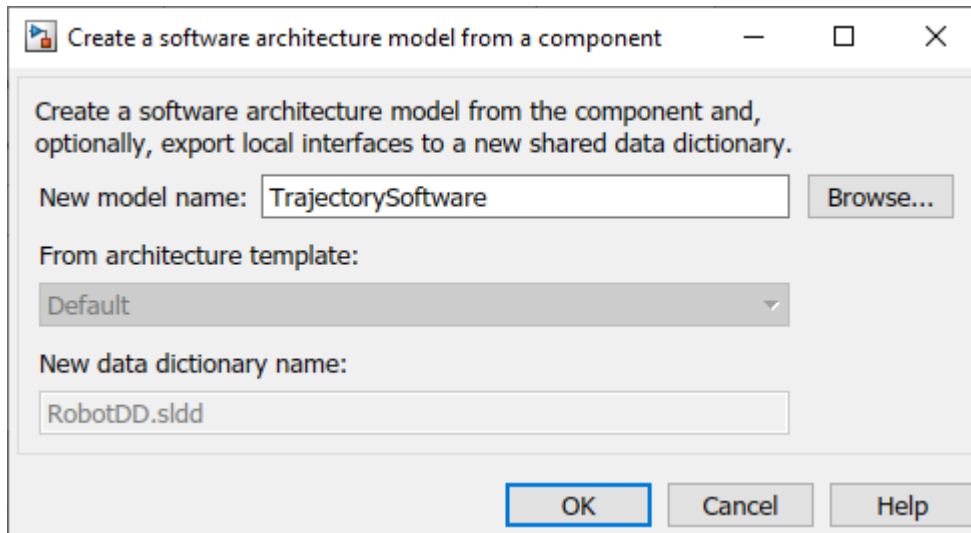
- 3 Select the SensorData output and view the interface in the Property Inspector. You can access the interface like a Simulink bus signal.



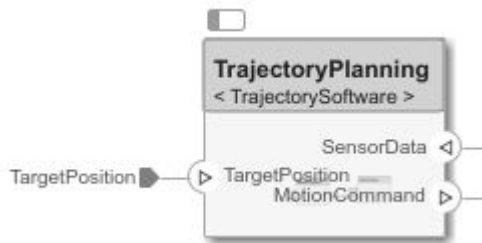
## Design Software Architecture in Component

To design a software architecture, define function execution order, simulate, and generate code, create a software architecture from a System Composer component.

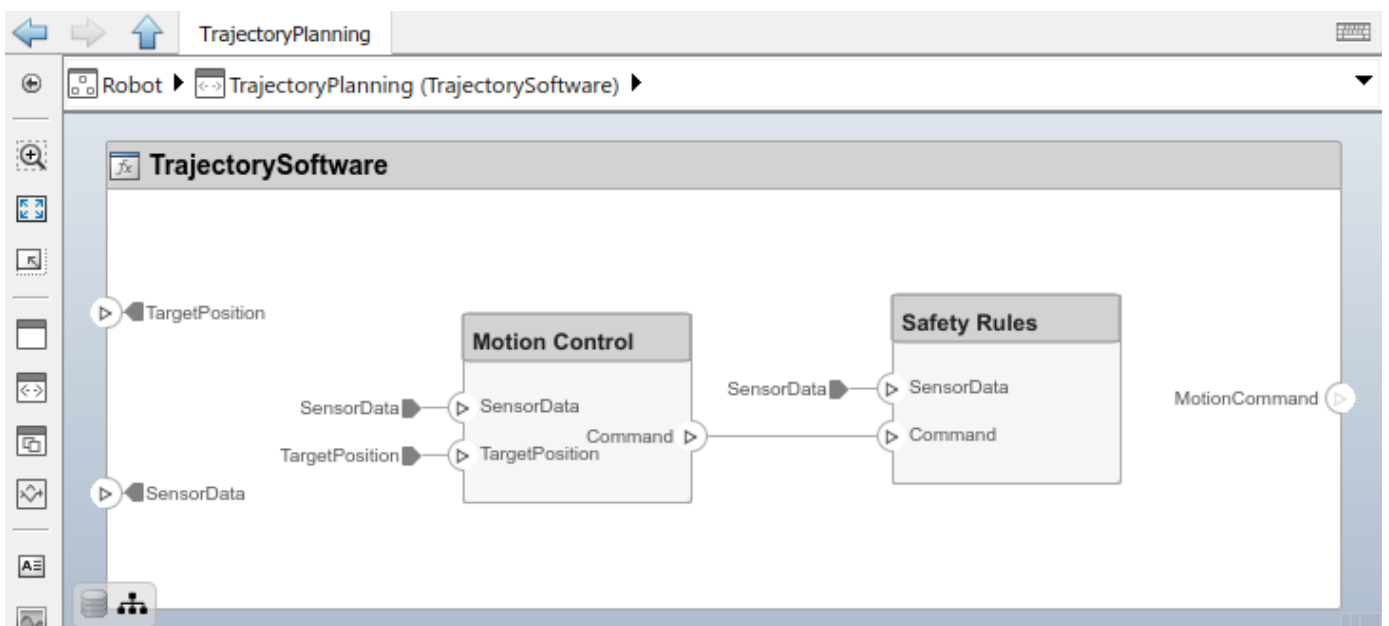
- 1 Rename the Trajectory Planning component to TrajectoryPlanning so that it is a valid C variable name.
- 2 Right-click the TrajectoryPlanning component and select Create Software Architecture Model, or, navigate to **Modeling > Component > Create Software Architecture Model**.
- 3 Specify the name of the software architecture as TrajectorySoftware. Click **OK**.



- 4 The software architecture model `TrajectorySoftware.slx` is referenced from the `TrajectoryPlanning` component.



- 5 Double-click the `TrajectoryPlanning` component to interact with the `TrajectorySoftware` software component.

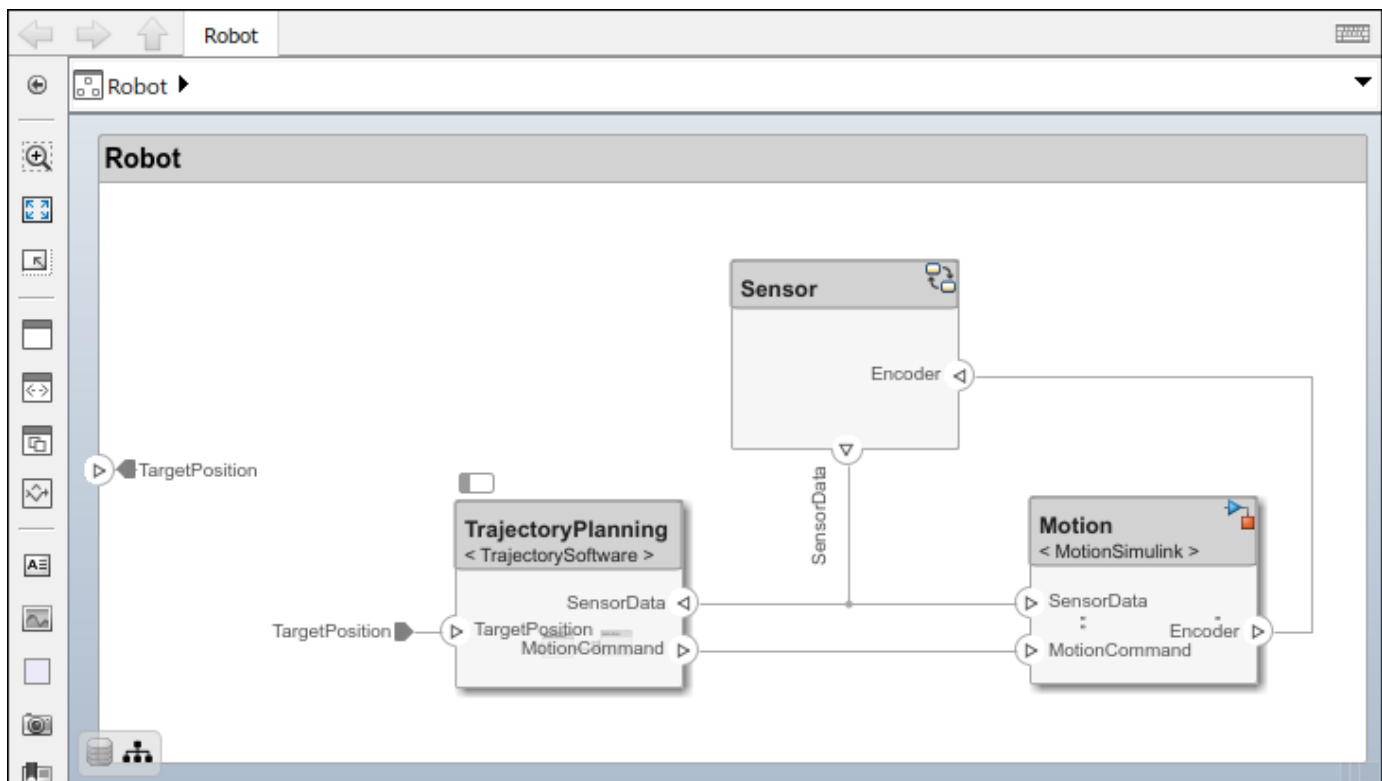




## Represent System Interaction Using Sequence Diagrams

To represent the interaction between structural elements of an architecture as a sequence of message exchanges, use a sequence diagram in the Architecture Views Gallery.

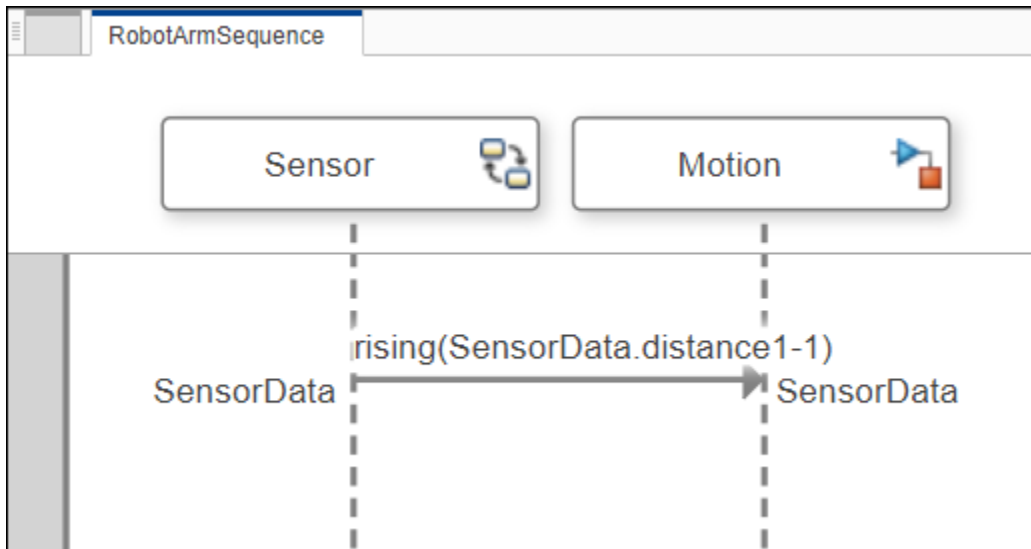
Observe the robot arm architecture model consisting of components, ports, connections, and behaviors. The model simulation results must match the interactions within the sequence diagrams.



- 1 Create a new sequence diagram by navigating to **Modeling > Views > Architecture Views**. The Architecture Views Gallery opens. To create a new sequence diagram, click **+ New > Sequence Diagram**.
- 2 A new sequence diagram called SequenceDiagram1 is created in the View Browser, and the **Sequence Diagram** tab becomes active. Under **Element Properties**, rename the sequence diagram RobotArmSequence.
- 3 Select **Component > Add Lifeline** **+** to add a lifeline. A new lifeline with no name is created and is indicated by a dotted line.
- 4 Click the down arrow and select Sensor. Add a second lifeline named Motion.
- 5 Select the vertical dotted line for the Sensor lifeline. Click and drag to the Motion lifeline. In the **To** box, start to type Sensordata and choose SensorData from the drop down menu. A message is created from the SensorData port on the Sensor component to the SensorData port on the Motion component.
- 6 Click on the message to see where to place the message condition. Enter a message trigger condition in the form:
 

```
rising(SensorData.distance1-1)
```

The signal name is a data element on a data interface. The message will be recognized at the zero-crossing event when the value of `SensorData.distance1` rises to 1.



### See Also

### More About

- "System Composer Concepts" on page 3-2

# System Composer Terminology

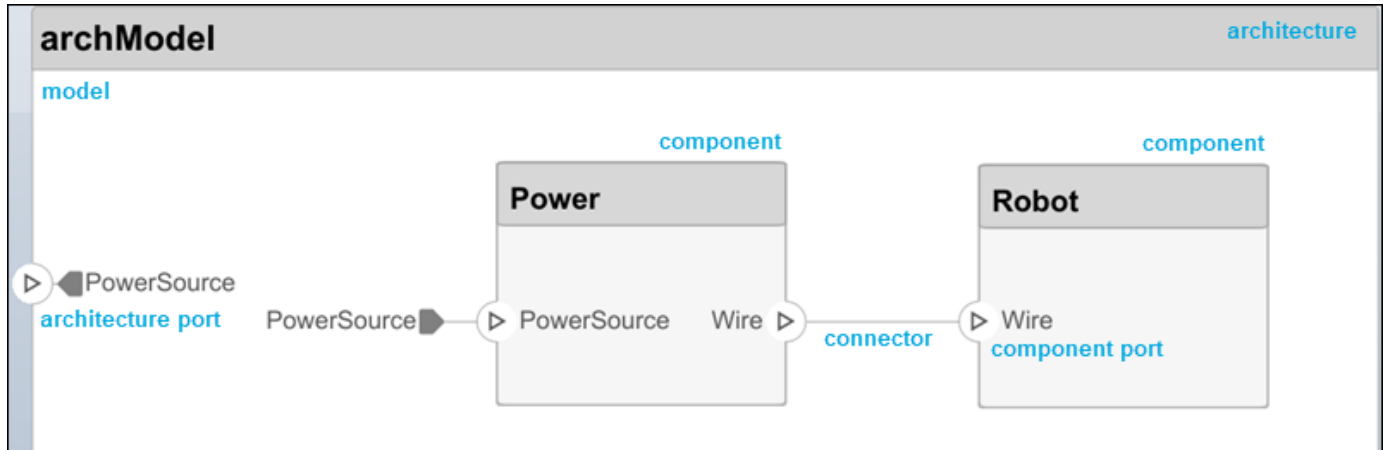
---

## System Composer Concepts

The terms in this topic provide a consistent and common language for using System Composer.

### Author Architecture Models

An architecture model in System Composer consists of the common Simulink constructions: components, ports, and connectors. An architecture represents the system of components.

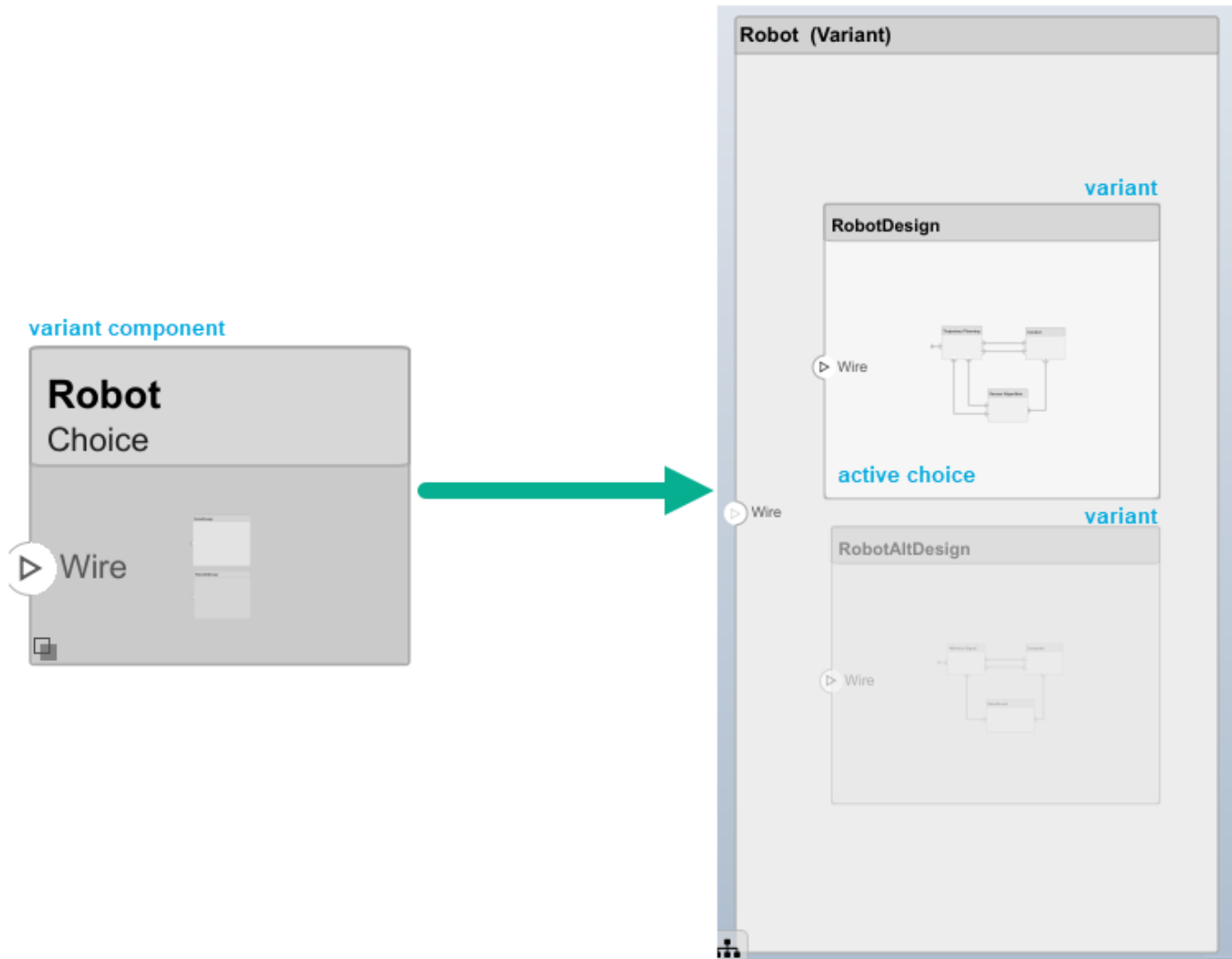


Term	Definition	Application	More Information
architecture	A System Composer architecture represents a system of components and how they interface with each other structurally and behaviorally. You can represent specific architectures using alternate views.	Different types of architectures describe different aspects of systems: <ul style="list-style-type: none"> <li>• <i>Functional architecture</i> describes the flow of data in a system.</li> <li>• <i>Logical architecture</i> describes the intended operation of a system.</li> <li>• <i>Physical architecture</i> describes the platform or hardware in a system.</li> </ul>	"Compose Architecture Visually"

Term	Definition	Application	More Information
model	A System Composer model is the file that contains architectural information, including components, ports, connectors, interfaces, and behaviors.	Perform operations on a model: <ul style="list-style-type: none"> <li>• Extract the root-level architecture contained in the model.</li> <li>• Apply profiles.</li> <li>• Link interface data dictionaries.</li> <li>• Generate instances from model architecture.</li> </ul> System Composer models are stored as SLX files.	“Create an Architecture Model with Interfaces and Requirement Links” on page 2-4
component	A component is a nontrivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of an architecture. A component defines an architecture element, such as a function, a system, hardware, software, or other conceptual entity. A component can also be a subsystem or subfunction.	Represented as a block, a component is a part of an architecture model that can be separated into reusable artifacts.	“Components”
port	A port is a node on a component or architecture that represents a point of interaction with its environment. A port permits the flow of information to and from other components or systems.	There are different types of ports: <ul style="list-style-type: none"> <li>• <i>Component ports</i> are interaction points on the component to other components.</li> <li>• <i>Architecture ports</i> are ports on the boundary of the system, whether the boundary is within a component or the overall architecture model.</li> </ul>	“Ports”
connector	Connectors are lines that provide connections between ports. Connectors describe how information flows between components or architectures.	A connector allows two components to interact without defining the nature of the interaction. Set an interface on a port to define how the components interact.	“Connections”

## Manage Variants

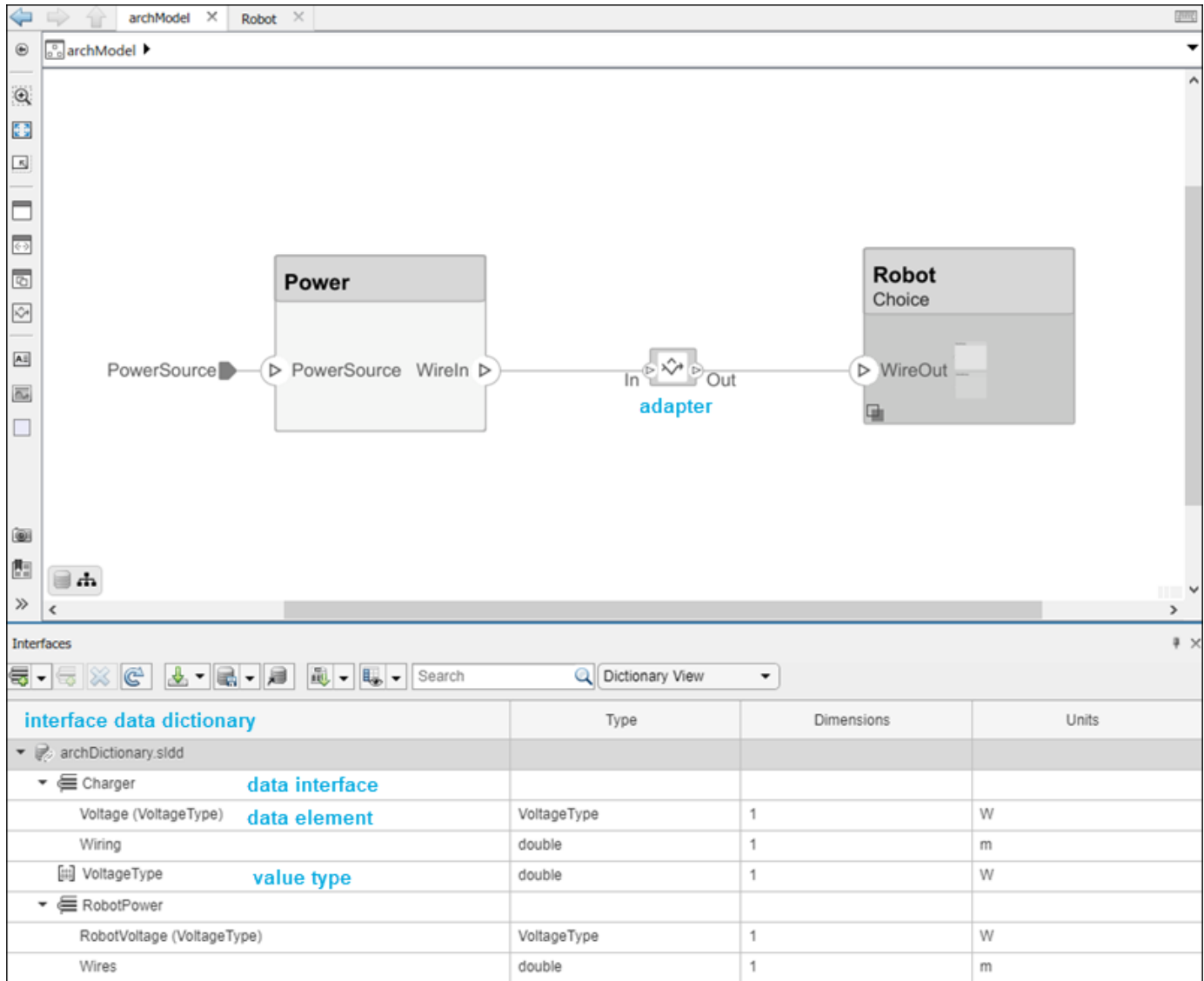
Create variant components and implement multiple design alternatives or variants, chosen based on programmatic rules. Add variant choices to any component to make a variant component. The active choice represents the original component.



Term	Definition	Application	More Information
variant	A variant is one of many structural or behavioral choices in a variant component.	Use variants to quickly swap different architectural designs for a component while performing analysis.	"Create Variants"
variant control	A variant control is a string that controls the active variant choice.	Set the variant control to programmatically control which variant is active.	"Set Condition"

## Manage Interfaces

Assign interfaces to ports using the Interface Editor in **Dictionary View**. Use an Adapter block to reconcile differences between interfaces on a connector between ports.

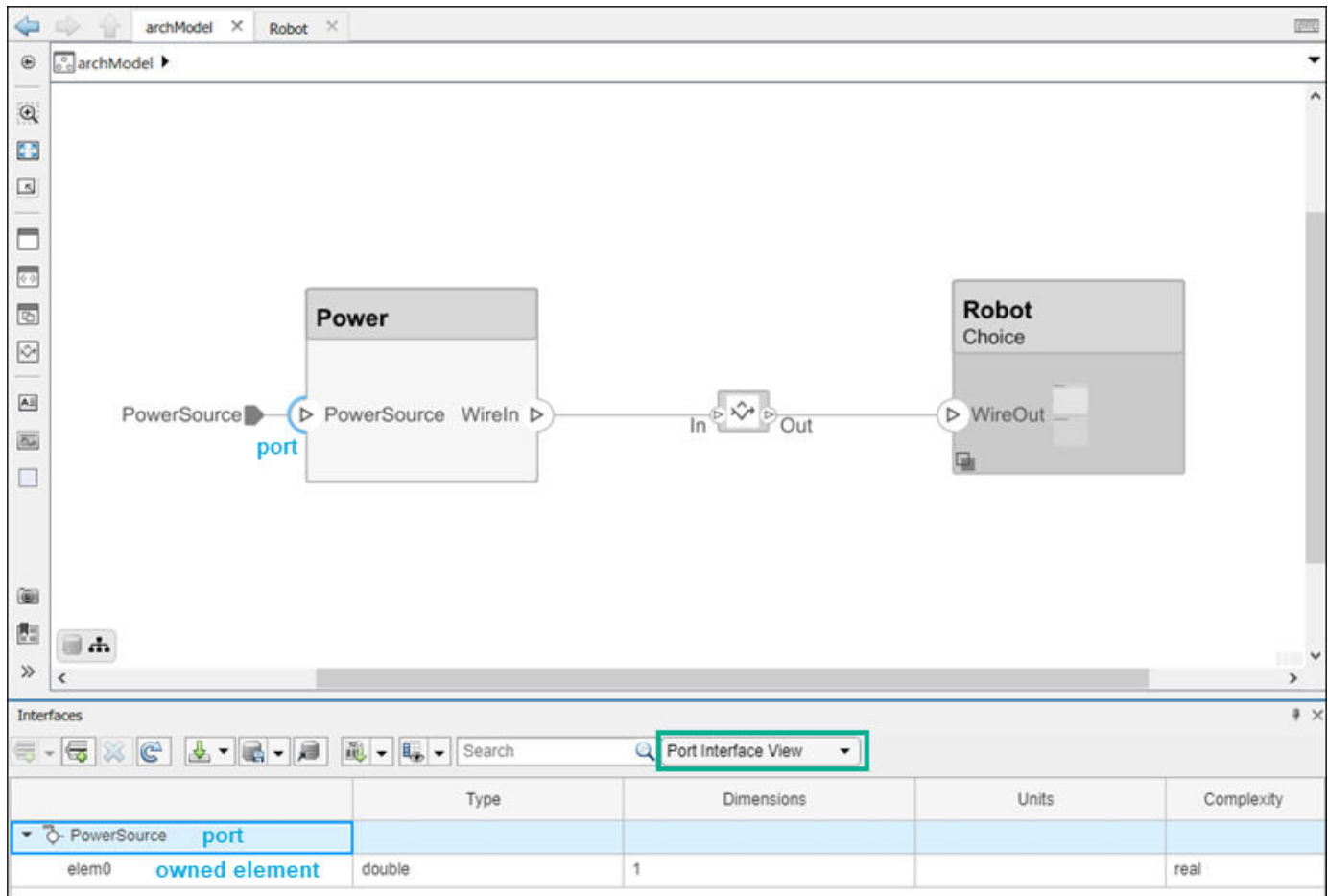


The diagram shows a 'Power' block with a 'PowerSource' port and a 'WireIn' port. A 'Robot' block has a 'WireOut' port. An 'adapter' block is connected between the 'WireIn' port of the 'Power' block and the 'WireOut' port of the 'Robot' block. The adapter block has an 'In' port and an 'Out' port.

The 'Interfaces' table below the diagram shows the dictionary for the archDictionary.sidd file:

interface data dictionary	Type	Dimensions	Units
archDictionary.sidd			
Charger	data interface		
Voltage (VoltageType)	data element	VoltageType	1 W
Wiring		double	1 m
VoltageType	value type	double	1 W
RobotPower			
RobotVoltage (VoltageType)		VoltageType	1 W
Wires		double	1 m

Manage interfaces local to a port using the Interface Editor in **Port Interface View**.



Term	Definition	Application	More Information
interface data dictionary	An interface data dictionary is a consolidated list of all the interfaces and value types in an architecture and where they are used.	Local interfaces on a System Composer model can be saved in an interface data dictionary using the Interface Editor.  Interface dictionaries can be reused between models that need to use a given set of interfaces, elements, and value types. Data dictionaries are stored in separate SLDD files.	<ul style="list-style-type: none"> <li>• “Manage Interfaces with Data Dictionaries”</li> <li>• “Reference Data Dictionaries”</li> </ul>

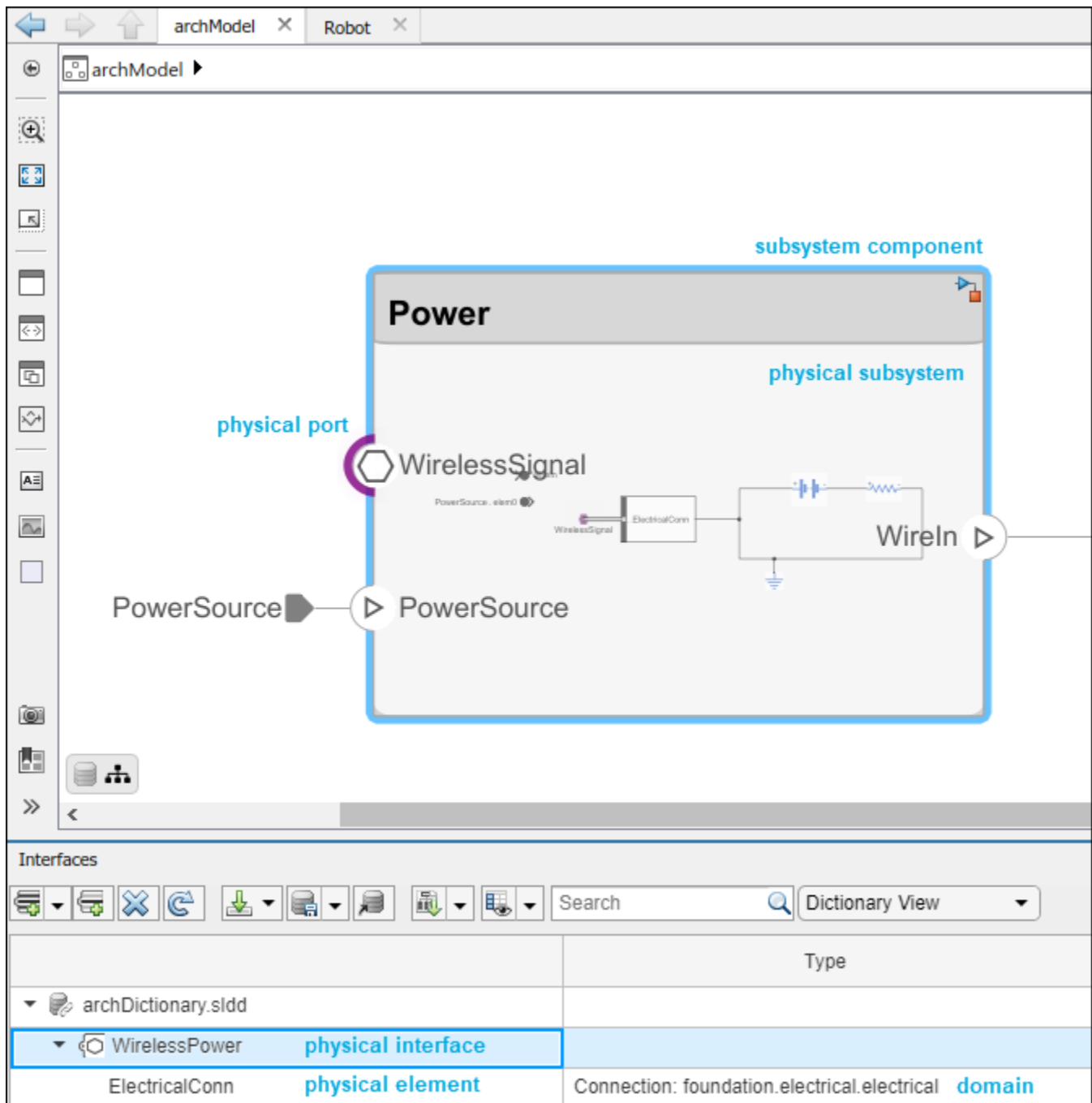


Term	Definition	Application	More Information
data interface	A data interface defines the kind of information that flows through a port. The same interface can be assigned to multiple ports. A data interface can be composite, meaning that it can include data elements that describe the properties of an interface signal.	Data interfaces represent the information that is shared through a connector and enters or exits a component through a port. Use the Interface Editor to create and manage data interfaces and data elements and store them in an interface data dictionary for reuse between models.	“Create an Architecture Model with Interfaces and Requirement Links” on page 2-4
data element	A data element describes a portion of an interface, such as a communication message, a calculated or measured parameter, or other decomposition of that interface.	Data interfaces are decomposed into data elements: <ul style="list-style-type: none"> <li>• Pins or wires in a connector or harness.</li> <li>• Messages transmitted across a bus.</li> <li>• Data structures shared between components.</li> </ul>	<ul style="list-style-type: none"> <li>• “Create Interfaces”</li> <li>• “Assign Interfaces to Ports”</li> </ul>
value type	A value type can be used as a port interface to define the atomic piece of data that flows through that port and 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. Add value types to data dictionaries using the Interface Editor so that you can reuse the value types as interfaces or data elements.	“Create Value Types as Interfaces”
owned interface	An owned interface is a locally defined interface that is local to a specific port and not shared in a data dictionary or the model dictionary.	Create an owned interface to represent a value type or data interface that is local to a port.	“Define Owned Interfaces Local to Ports”

Term	Definition	Application	More Information
adapter	An adapter helps connect two components with incompatible port interfaces by mapping between the two interfaces. An adapter can also act as a unit delay or rate transition. Use the Adapter block to implement an adapter.	<p>With an adapter, you can perform functions on the Interface Adapter dialog:</p> <ul style="list-style-type: none"> <li>• Create and edit mappings between input and output interfaces.</li> <li>• Apply an interface conversion <code>UnitDelay</code> to break an algebraic loop.</li> <li>• Apply an interface conversion <code>RateTransition</code> to reconcile different sample time rates for reference models.</li> </ul>	<ul style="list-style-type: none"> <li>• “Interface Adapter”</li> <li>• Adapter</li> </ul>

### Author Physical Models

Author physical models in System Composer using subsystem components. A subsystem component is a Simulink subsystem that is part of the parent System Composer architecture model. You can add Simscape behavior to a subsystem component using physical ports, connectors, and interfaces.

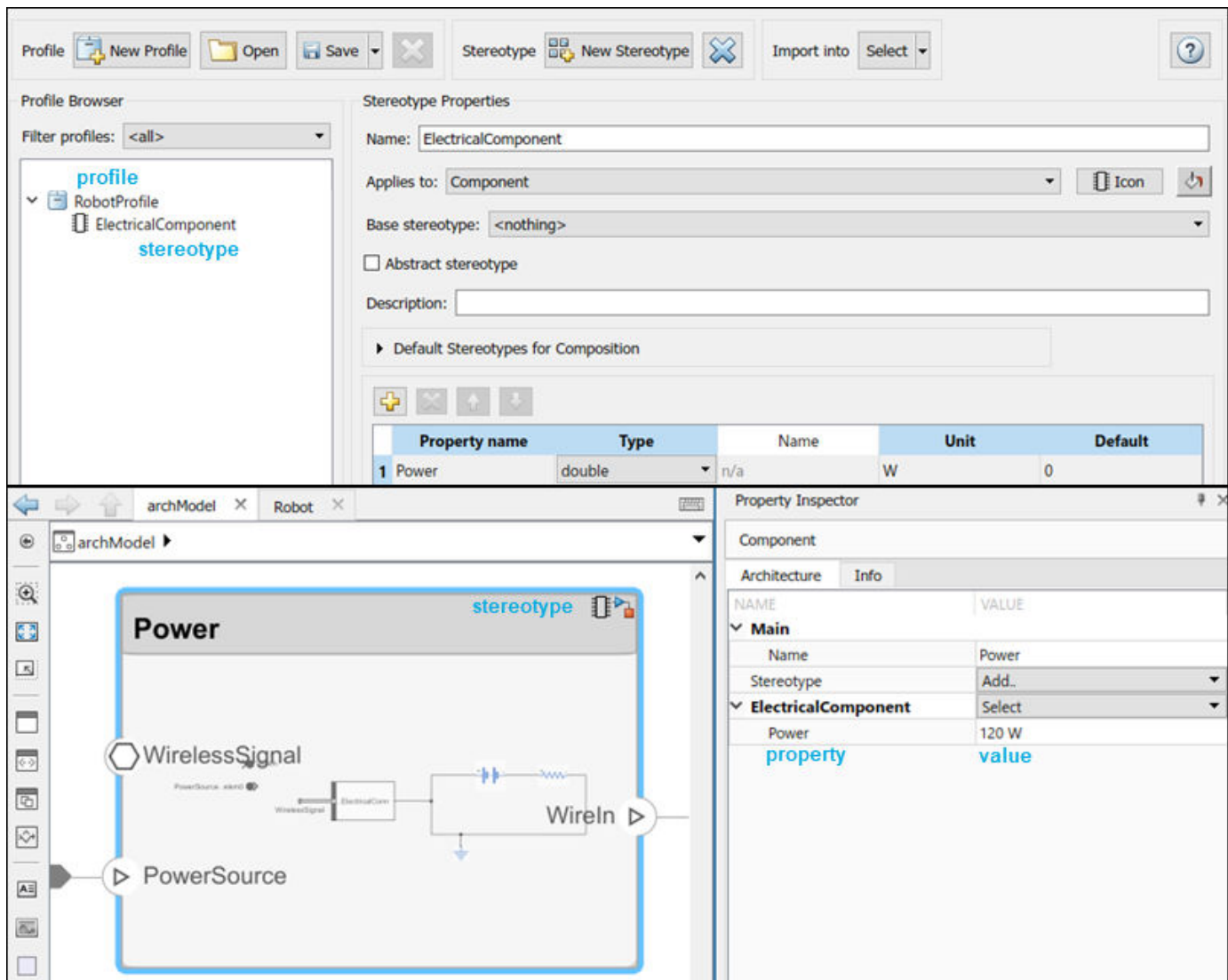


For more information, see “Author Model Behavior” on page 3-20.

Term	Definition	Application	More Information
physical subsystem	A physical subsystem is a Simulink subsystem with Simscape connections.	A physical subsystem with Simscape connections uses a physical network approach suited for simulating systems with real physical components and represents a mathematical model.	"Describe Component Behavior Using Simscape"
physical port	A physical port represents a Simscape physical modeling connector port called a Connection Port.	Use physical ports to connect components in an architecture model or to enable physical systems in a Simulink subsystem.	"Define Physical Ports on a Component"
physical connector	A physical connector can represent a nondirectional conserving connection of a specific physical domain. Connectors can also represent physical signals.	Use physical connectors to connect physical components that represent features of a system to simulate mathematically.	"Architecture Model with Simscape Behavior for a DC Motor"
physical interface	A physical interface defines the kind of information that flows through a physical port. The same interface can be assigned to multiple ports. A physical interface is a composite interface equivalent to a <code>Simulink.ConnectionBus</code> object that specifies at least one <code>Simulink.ConnectionElement</code> object.	Use a physical interface to bundle physical elements to describe a physical model using at least one physical domain.	"Specify Physical Interfaces on the Ports"
physical element	A physical element describes the decomposition of a physical interface. A physical element is equivalent to a <code>Simulink.ConnectionElement</code> object.	Define the Type of a physical element as a physical domain to enable use of that domain in a physical model.	"Describe Component Behavior Using Simscape"

### Extend Architectural Elements

Create a profile in the Profile Editor and add stereotypes to it with properties. Apply the stereotype to a component, and set the property value.

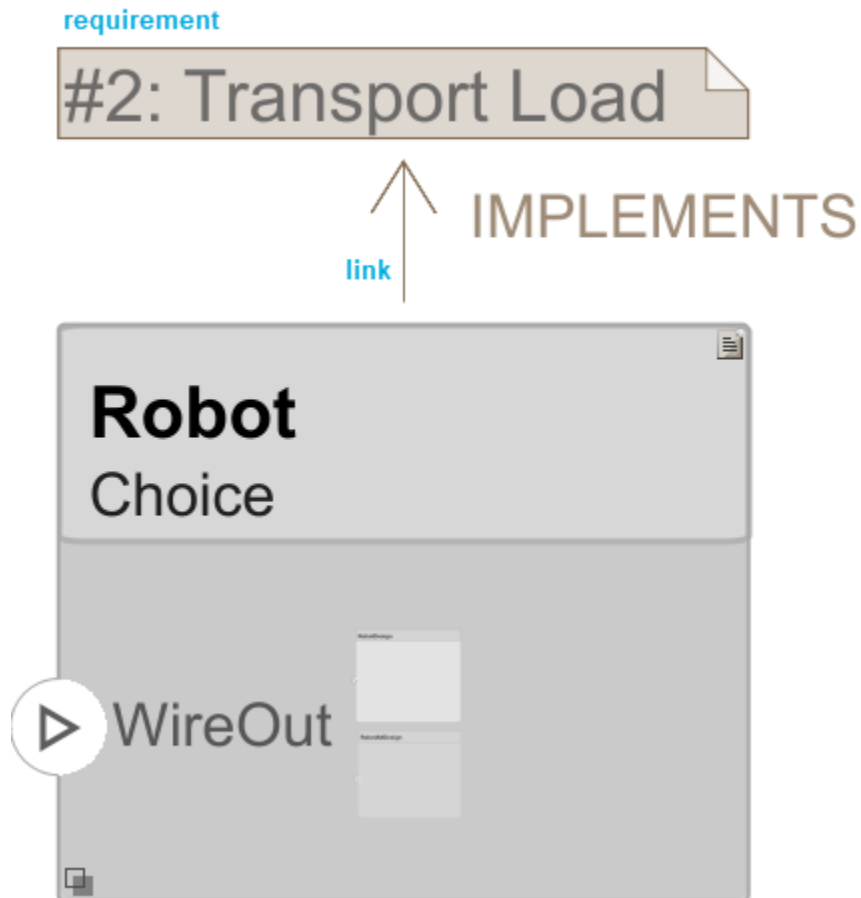


Term	Definition	Application	More Information
stereotype	A stereotype is a custom extension of the modeling language. Stereotypes provide a mechanism to extend the architecture language elements by adding domain-specific metadata.	Apply stereotypes to elements: root-level architecture, component architecture, connectors, ports, data interfaces, and value types of a model. A model element can have multiple stereotypes. Stereotypes provide model elements with a common set of property fields, such as mass, cost, and power.	"Extend Architectural Design Using Stereotypes" on page 2-18

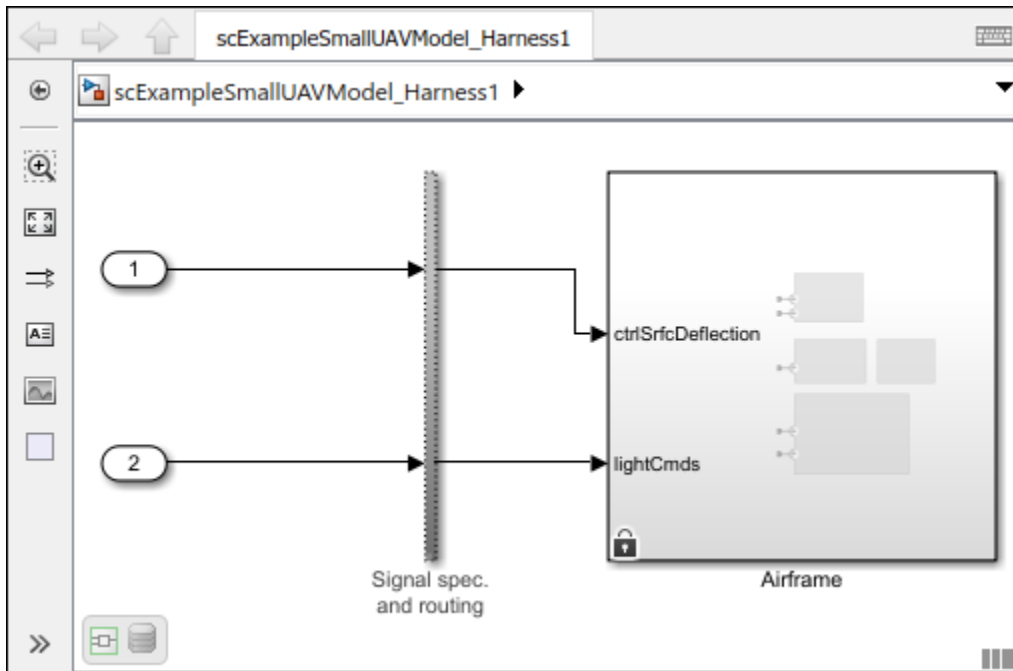
Term	Definition	Application	More Information
property	A property is a field in a stereotype. For each element the stereotype is applied to, specific property values are specified.	Use properties to store quantitative characteristics, such as weight or speed, that are associated with a model element. Properties can also be descriptive or represent a status. You can view and edit the properties of each element in the architecture model using the Property Inspector.	<ul style="list-style-type: none"> <li>• “Set Properties” on page 2-23</li> <li>• “Add Properties with Stereotypes”</li> <li>• “Set Properties for Analysis”</li> </ul>
profile	A profile is a package of stereotypes to create a self-consistent domain of element types.	Author profiles and apply profiles to a model using the Profile Editor. You can store stereotypes for a project in one profile or in several. Profiles are stored in XML files when they are saved.	<ul style="list-style-type: none"> <li>• “Define Profiles and Stereotypes”</li> <li>• “Use Stereotypes and Profiles”</li> </ul>

## Manage and Verify Requirements

In the Requirements perspective, you can create, manage, and allocate requirements. View the requirements on the architecture model. This functionality requires a Simulink Requirements license.



Use Simulink Test to create a test harness for a System Composer component to validate simulation results and verify design in the Test Manager. This functionality requires a Simulink Test license.



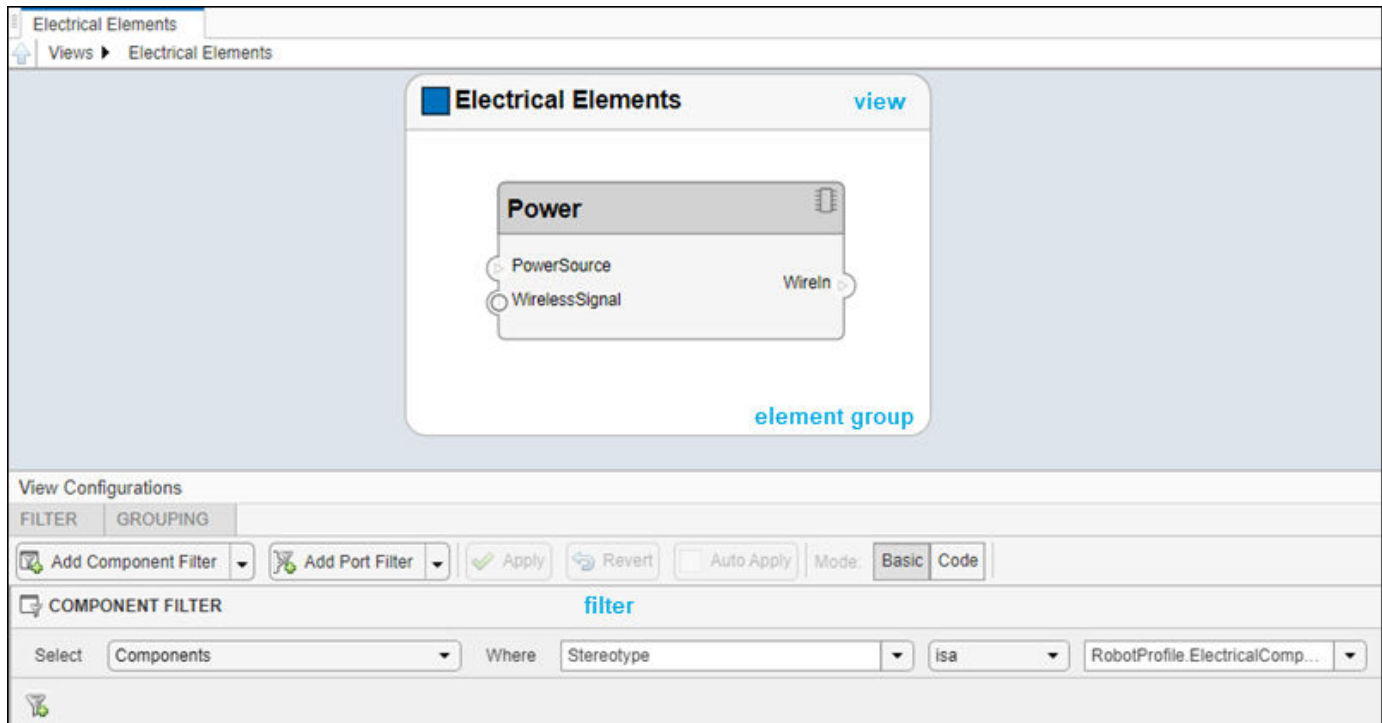
Term	Definition	Application	More Information
requirements	Requirements are a collection of statements describing the desired behavior and characteristics of a system. Requirements ensure system design integrity and are achievable, verifiable, unambiguous, and consistent with each other. Each level of design should have appropriate requirements.	To enhance traceability of requirements, link system, functional, customer, performance, or design requirements to components and ports. Link requirements to each other to represent derived or allocated requirements. Manage requirements from the Requirements Manager on an architecture model or through custom views. Assign test cases to requirements using the Test Manager for verification and validation.	“Link and Trace Requirements”



Term	Definition	Application	More Information
requirement set	A requirement set is a collection of requirements. You can structure the requirements hierarchically and link them to components or ports.	Use the Requirements Editor to edit and refine requirements in a requirement set. Requirement sets are stored in <code>.slreqx</code> files. You can create a new requirement set and author requirements using Simulink Requirements, or import requirements from supported third-party tools.	“Manage Requirements”
requirement link	A link is an object that relates two model-based design elements. A requirement link is a link where the destination is a requirement. You can link requirements to components or ports.	View links using the Requirements perspective in System Composer. Select a requirement in the Requirements Browser to highlight the component or the port to which the requirement is assigned. Links are stored externally as <code>.slmx</code> files.	<ul style="list-style-type: none"> <li>• “Create an Architecture Model with Interfaces and Requirement Links” on page 2-4</li> <li>• “Update Reference Requirement Links from Imported File”</li> </ul>
test harness	A test harness is a model that isolates the component under test, with inputs, outputs, and verification blocks configured for testing scenarios. You can create a test harness for a model component or for a full model. A test harness gives you a separate testing environment for a model or a model component.	Create a test harness for a System Composer component to validate simulation results and verify design. The Interface Editor is accessible in System Composer test harness models to enable behavior testing and implementation-independent interface testing.	<ul style="list-style-type: none"> <li>• “Verify and Validate Requirements Using Test Harnesses on Components”</li> <li>• “Create a Test Harness” (Simulink Test)</li> </ul>

## Create Custom Views

Apply a view filter to generate an element group of components for the view in the Architecture Views Gallery.

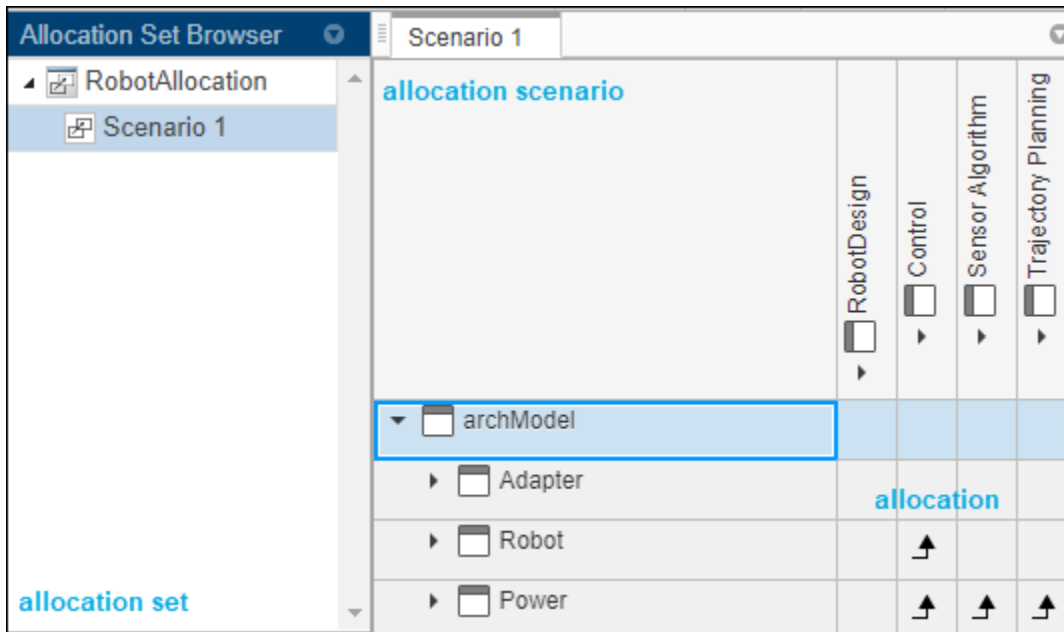


Term	Definition	Application	More Information
view	A view shows a customizable subset of elements in a model. Views can be filtered based on stereotypes or names of components, ports, and interfaces, along with the name, type, or units of an interface element. Create views by adding elements manually. Views create a simplified way to work with complex architectures by focusing on certain parts of the architecture design.	<p>You can use different types of views to represent the system:</p> <ul style="list-style-type: none"> <li>• <i>Operational views</i> demonstrate how a system will be used and should be integrated with requirements analysis.</li> <li>• <i>Functional views</i> focus on what the system must do to operate.</li> <li>• <i>Physical views</i> show how the system is constructed and configured.</li> </ul> <p>A viewpoint represents a stakeholder perspective that specifies the contents of the view.</p>	"Modeling System Architecture of Keyless Entry System"

Term	Definition	Application	More Information
element group	An element group is a grouping of components in a view.	Use element groups to programmatically populate a view.	<ul style="list-style-type: none"> <li>• “Create Architecture Views Interactively”</li> <li>• “Create Architectural Views Programmatically”</li> </ul>
query	A query is a specification that describes certain constraints or criteria to be satisfied by model elements.	Use queries to search elements with constraint criteria and to filter views.	“Find Elements in Model Using Queries”
component diagram	A component diagram represents a view with components, ports, and connectors based on how the model is structured.	Component diagrams allow you to programmatically or manually add and remove components from the view.	“Inspect Components in Custom Architecture Views” on page 2-31
hierarchy diagram	You can visualize a hierarchy diagram as a view with components, ports, reference types, component stereotypes, and stereotype properties.	<p>There are two types of hierarchy diagrams:</p> <ul style="list-style-type: none"> <li>• <i>Component hierarchy diagrams</i> display components in tree form with parents above children. In a component hierarchy view, each referenced model is represented as many times as it is used.</li> <li>• <i>Architecture hierarchy diagrams</i> display unique component architecture types and their relationships using composition connections. In an architecture hierarchy view, each referenced model is represented only once.</li> </ul>	“Display Component Hierarchy and Architecture Hierarchy Using Views”

## Allocate Architecture Models

In the Allocation Editor, allocate components between two architecture models, based on a dependency or a directed relationship.



Term	Definition	Application	More Information
allocation	An allocation is a directed relationship from an element in one model to an element in another model.	Resource-based allocation allows you to allocate functional architectural elements to logical architectural elements and logical architectural elements to physical architectural elements.	“Allocate Architectures in Tire Pressure Monitoring System”
allocation scenario	An allocation scenario contains a set of allocations between a source and target model.	Allocate between model elements within an allocation in an allocation scenario. The default allocation scenario is called Scenario 1.	“Create and Manage Allocations”
allocation set	An allocation set consists of one more allocation scenarios which describe various allocations between a source and target model.	Create an allocation set with allocation scenarios.	“Create and Manage Allocations”

## Analyze Architecture Models

Create an analysis function to analyze power consumption in the RobotDesign architecture model.

```
function robotDesign(instance, varargin)
if instance.isComponent()
    if instance.hasValue('RobotProfile.ElectricalComponent.Power')
        sysComponent_power = instance.getValue('RobotProfile.ElectricalComponent.Power');
    end
    for child = instance.Components
        comp_power = child.getValue('RobotProfile.ElectricalComponent.Power');
```

```

        sysComponent_power = sysComponent_power + comp_power;
    end
    instance.setValue('RobotProfile.ElectricalComponent.Power', sysComponent_power);
end

```

Analyze the robot design using the analysis function to determine total power usage.

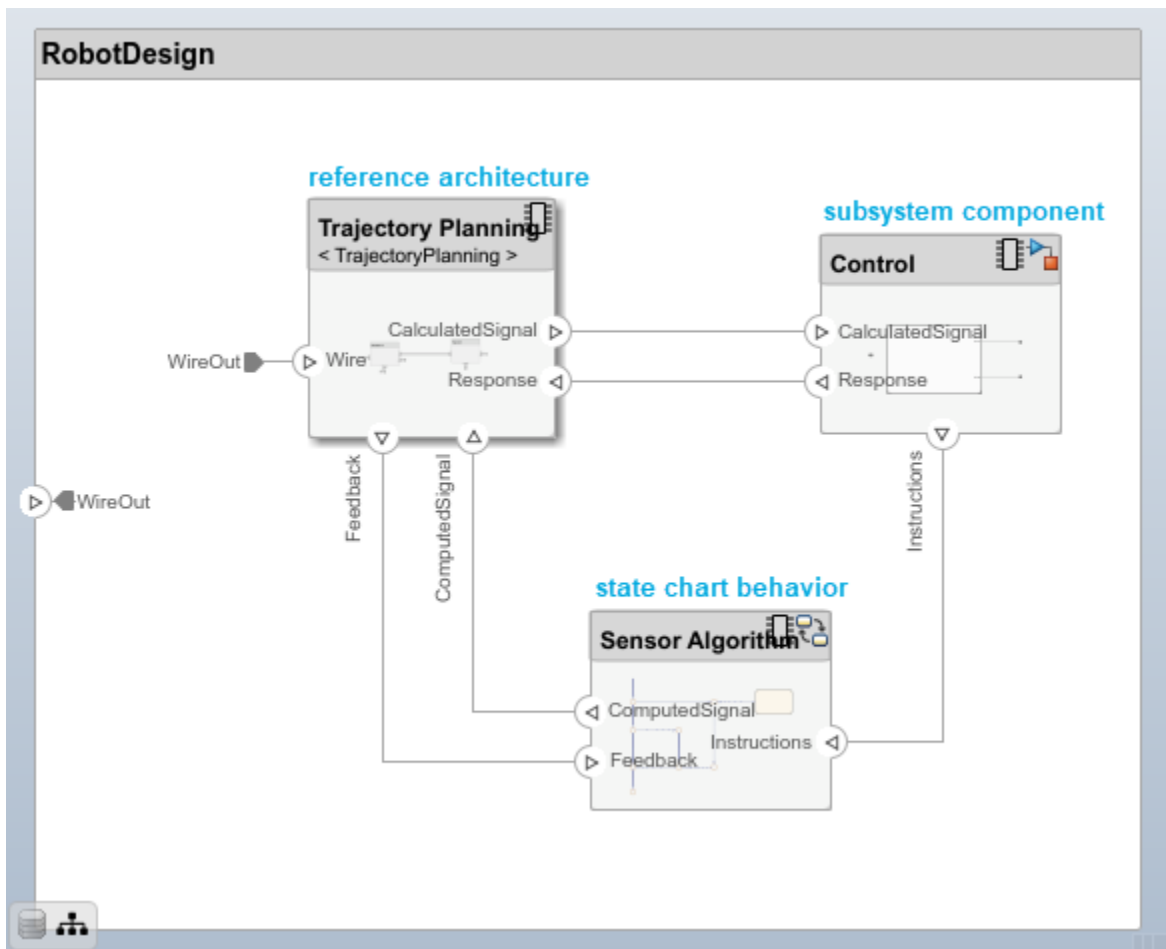
The screenshot displays the System Composer software interface. At the top, there is a 'HOME' tab and a toolbar with icons for 'New', 'Open', 'Save', 'Delete', 'Analyze', 'Refresh', and 'Update'. The 'Analyze' button is highlighted. Below the toolbar, there are sections for 'INSTANCE MODEL', 'ANALYSIS', 'REFRESH', and 'UPDATE'. The 'ANALYSIS' section shows a dropdown menu set to 'BottomUp'. The 'INSTANCE MODEL' section shows a tree view with 'RobotDesign' selected, containing sub-items: 'Control' (50), 'Sensor Algorithm' (29), and 'Trajectory Planning' (90). The 'ANALYSIS' section shows a table with 'Power' and '169'. The 'REFRESH' section shows 'Automatic' and 'Overwrite' checkboxes. The 'UPDATE' section shows 'Update' and 'Refresh' buttons. The 'INSTANCE PROPERTIES' panel on the right shows 'ComponentInstance: RobotDesign' and a table with 'Property', 'Value', and 'Units' columns. The table shows 'RobotProfile.ElectricalComp' with 'Power' set to '169 W' and 'analysis' in blue text.

Term	Definition	Application	More Information
analysis	Analysis is a method for quantitatively evaluating an architecture for certain characteristics. Static analysis analyzes the structure of the system. Static analysis uses an analysis function and parametric values of properties captured in the system model.	Use analyses to calculate overall reliability, mass roll-up, performance, or thermal characteristics of a system, or to perform a SWaP analysis.	<ul style="list-style-type: none"> <li>“Analyze an Architecture Model with an Analysis Function” on page 2-27</li> <li>“Analyze Architecture”</li> </ul>
analysis function	An analysis function is a MATLAB function that computes values necessary to evaluate the architecture using properties of each element in the model instance.	Use an analysis function to calculate the result of an analysis.	“Write Analysis Function”

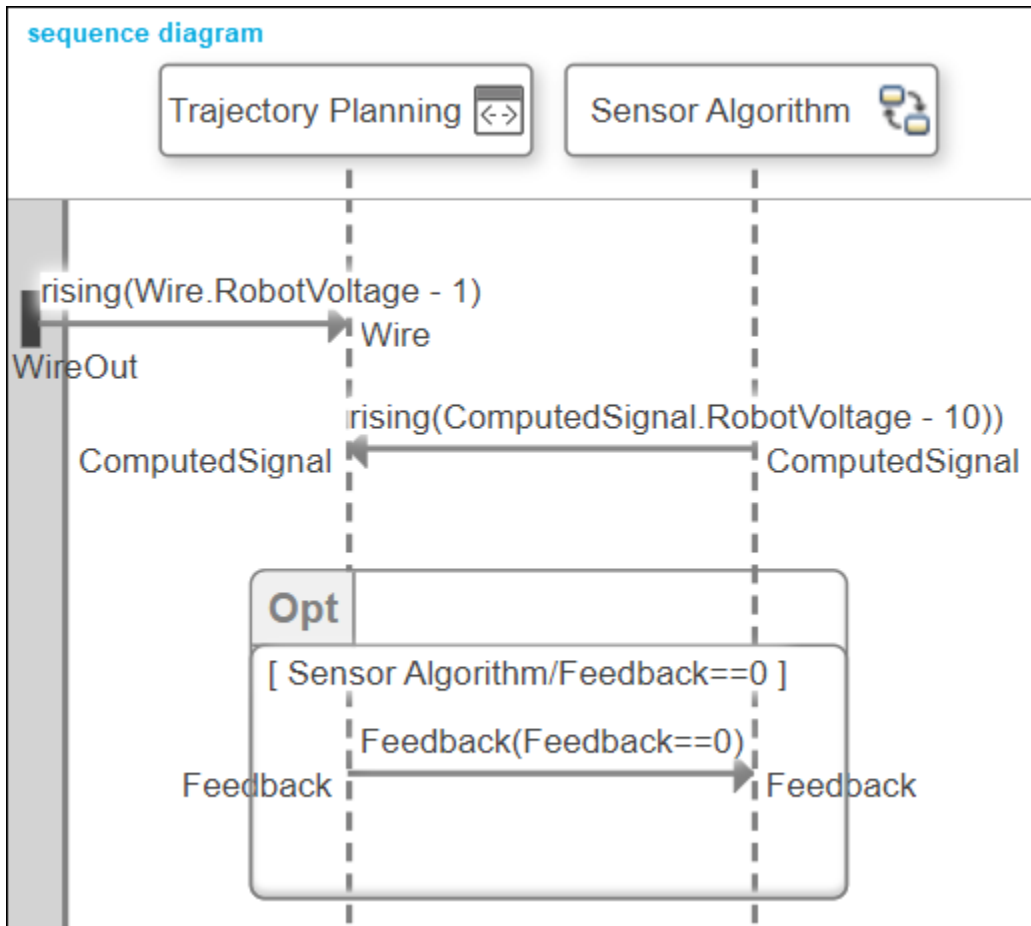
<b>Term</b>	<b>Definition</b>	<b>Application</b>	<b>More Information</b>
instance model	An instance model is a collection of instances.	You can update an instance model with changes to a model, but the instance model will not update with changes in active variants or model references. You can use an instance model, saved in an .MAT file, of a System Composer architecture model for analysis.	“Run Analysis Function”
instance	An instance is an occurrence of an architecture model element at a given point in time.	An instance freezes the active variant or model reference of the component in the instance model.	“Create a Model Instance for Analysis”

### **Author Model Behavior**

Use a reference component to decompose and reuse architectural components and Simulink model behaviors. Use a subsystem component or state chart to implement Simulink and Stateflow behaviors.



Create a sequence diagram in the Architecture Views Gallery to describe system interactions.



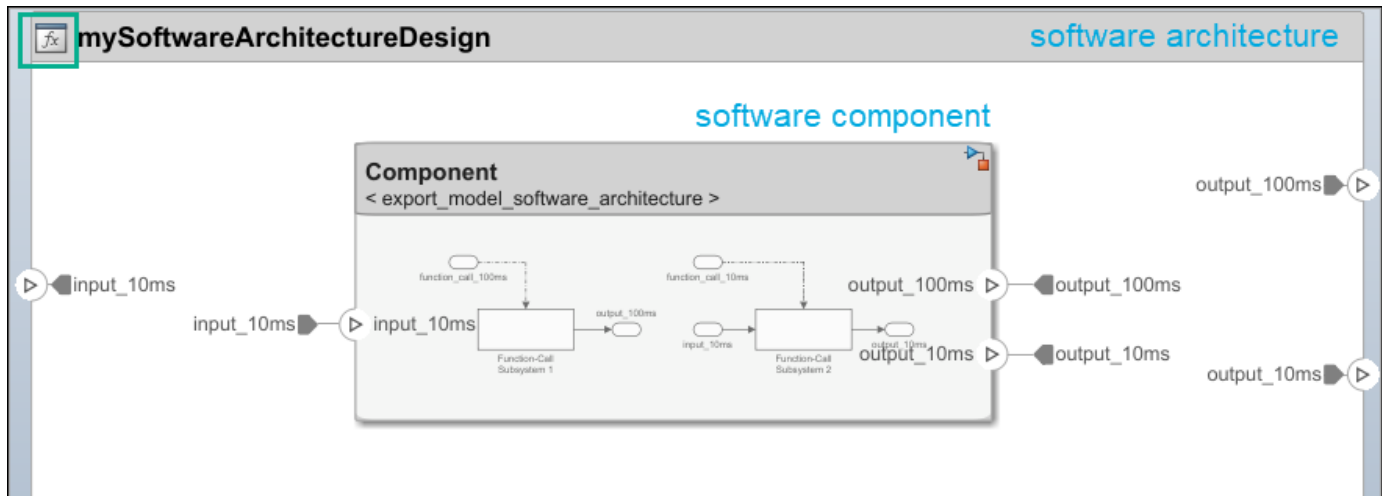
Term	Definition	Application	More Information
reference component	A reference component is a component whose definition is a separate architecture model or Simulink behavior model.	A reference component represents a logical hierarchy of other compositions. You can reuse compositions in the model using reference components.	<ul style="list-style-type: none"> <li>• “Describe Component Behavior Using Simulink”</li> <li>• “Create Reference Architecture”</li> </ul>
subsystem component	A subsystem component is a Simulink subsystem that is part of the parent System Composer architecture model.	Add Simulink subsystem behavior to a component to author a subsystem component in System Composer. You cannot synchronize and reuse subsystem components as Reference Component blocks because the component is part of the parent model.	<ul style="list-style-type: none"> <li>• “Create Simulink Behavior Using Simulink Subsystem”</li> <li>• “Create a Simulink Subsystem Component”</li> </ul>



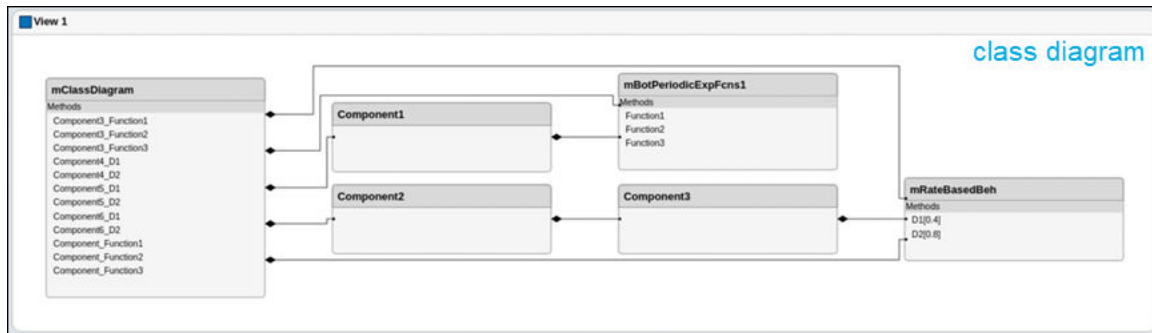
Term	Definition	Application	More Information
state chart	A state chart diagram demonstrates the state-dependent behavior of a component throughout its state lifecycle and the events that can trigger a transition between states.	Add Stateflow chart behavior to describe a component using state machines. You cannot synchronize and reuse Stateflow chart behaviors as Reference Component blocks because the component is part of the parent model.	<ul style="list-style-type: none"> <li>“Implement Behaviors for Architecture Model Simulation” on page 2-37</li> <li>“Describe Component Behavior Using Stateflow Charts”</li> </ul>
sequence diagram	A sequence diagram is a behavior diagram that represents the interaction between structural elements of an architecture as a sequence of message exchanges.	You can use sequence diagrams to describe how the parts of a static system interact.	<ul style="list-style-type: none"> <li>“Describe System Behavior Using Sequence Diagrams”</li> <li>“Use Sequence Diagrams with Architecture Models”</li> </ul>

## Design Software Architectures

Design a software architecture model, define the execution order of the functions from the components, simulate the design in the architecture level, and generate code.



View the software architecture diagram in a class diagram in the Architecture Views Gallery.



Term	Definition	Application	More Information
software architecture	A software architecture is a specialization of an architecture for software-based systems, including the description of software compositions, component functions, and their scheduling	Use software architectures in System Composer to author software architecture models composed of software components, ports, and interfaces. Design your software architecture model, define the execution order of your component functions, simulate your design in the architecture level, and generate code.	<ul style="list-style-type: none"> <li>“Author Software Architectures”</li> <li>“Simulate and Deploy Software Architectures”</li> </ul>
software component	A software component is a specialization of a component for software entities, including its functions (entry points) and interfaces.	Implement a Simulink export-function, rate-based, or JMAAB model as a software component, simulate the software architecture model, and generate code.	<ul style="list-style-type: none"> <li>“Implement Behaviors for Architecture Model Simulation” on page 2-37</li> <li>“Create Software Architecture from Component”</li> </ul>
software composition	A software composition is a diagram of software components and connectors that represents a composite software entity, such as a module or application.	Encapsulate functionality by aggregating or nesting multiple software components or compositions.	“Modeling the Software Architecture of a Throttle Position Control System”
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. Any component diagram view can be optionally represented as a class diagram for a software architecture model.	“Class Diagram View of Software Architectures”

## See Also

### More About

- “Compose and Analyze a System Using an Architecture Model” on page 2-2
- “Organize System Composer Files in a Project”
- “Simulate Mobile Robot with System Composer Workflow”
- “Modeling System Architecture of Small UAV”

### External Websites

- <https://www.mathworks.com/videos/series/systems-engineering.html>

