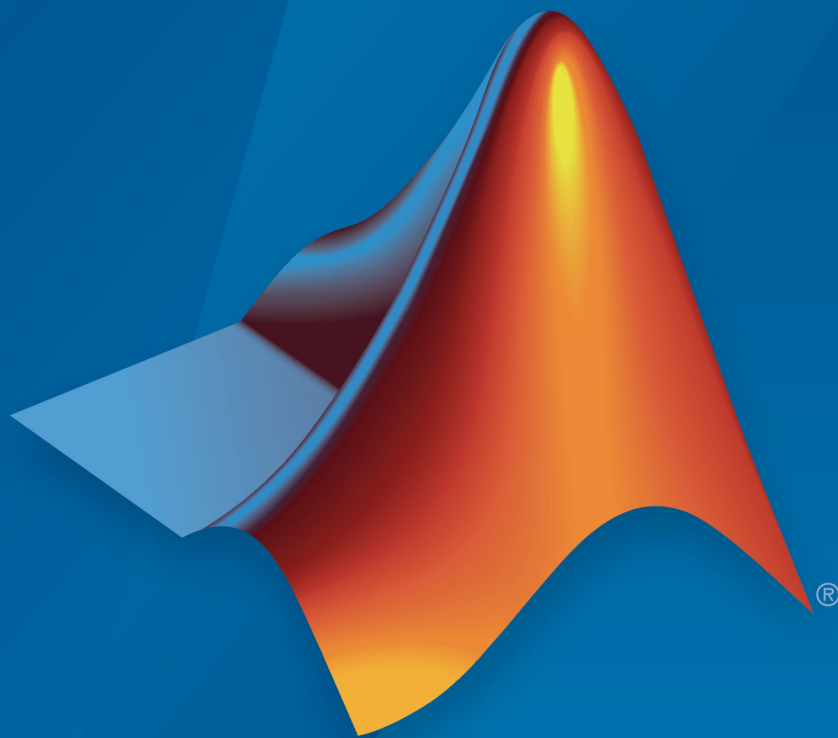


# System Composer™

## Getting Started Guide



# MATLAB® & SIMULINK®

R2019b



# 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 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)

## **1** Compose an Architecture Model

<b>Compose and Analyze a System</b> .....	<b>1-2</b>
<b>Create an Architecture Model</b> .....	<b>1-6</b>
Visually Represent the System .....	<b>1-6</b>
Edit Interfaces .....	<b>1-14</b>
Decompose Components .....	<b>1-17</b>
Implement Component Behavior .....	<b>1-19</b>
Link to an Existing Simulink Behavior Model .....	<b>1-22</b>
<b>Inspect Components in Custom Views</b> .....	<b>1-23</b>
<b>Analyze an Architecture Model</b> .....	<b>1-26</b>
Load Architecture Model Profile .....	<b>1-27</b>
Apply Stereotypes to Model Elements .....	<b>1-29</b>
Set Properties .....	<b>1-32</b>
Perform an Analysis .....	<b>1-35</b>

## **2** Refactor a Simulink Model as a Composition

<b>Implement Component Behavior in Simulink</b> .....	<b>2-2</b>
Create a Simulink Behavior Model .....	<b>2-2</b>
Link to an Existing Simulink Behavior Model .....	<b>2-4</b>
<b>Export Simulink Model as Architecture</b> .....	<b>2-6</b>



# Compose an Architecture Model

---

- “Compose and Analyze a System” on page 1-2
- “Create an Architecture Model” on page 1-6
- “Inspect Components in Custom Views” on page 1-23
- “Analyze an Architecture Model” on page 1-26

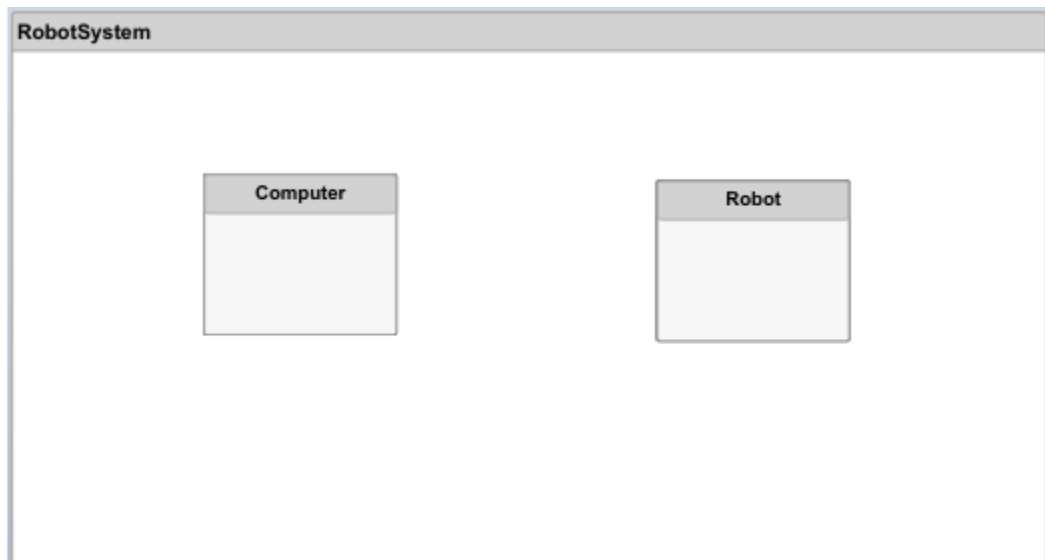
## Compose and Analyze a System

A system is a composition of different elements that serves a goal that cannot be achieved by any of the elements alone. The elements of a system can be mechanical parts, electrical circuits, computer hardware, and software. A system solution consists of a set of elements as well as their characteristics and properties. System Composer enables you to create architecture models using structural and behavioral diagrams that all act on the same underlying model. This way, you ensure that a change in one diagram is reflected in the other diagrams, resulting in a consistent system model that you can share.

With System Composer, you can:

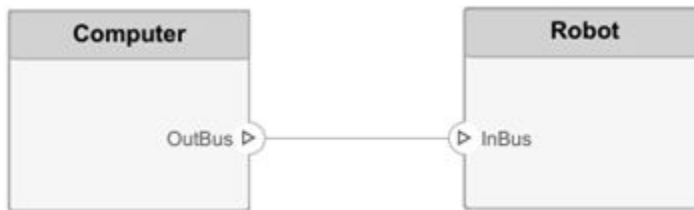
- Create structural models using hierarchical functional, logical, and physical architecture diagrams
- Support specific architectural requirements by customizing architectural types
- Validate behavior, and refine and elaborate requirements
- Perform static analysis and trade studies to optimize system architectures

Consider a mobile robotic system where a computer sends a target location to the robot wirelessly. This system has two primary components: the computer and the robot. You represent them in System Composer using two Component blocks.



To capture specifications relevant to the problem, you can add non-visual properties to a component. For example, if the total power consumption of the system is a concern, a **Power Consumption** property is necessary. You can add this property to an electrical component using a stereotype. A stereotype adds properties to components, ports, and connectors.

Connections are essential in describing a system as a network of components. In System Composer, you define ports on each component and connect them.



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

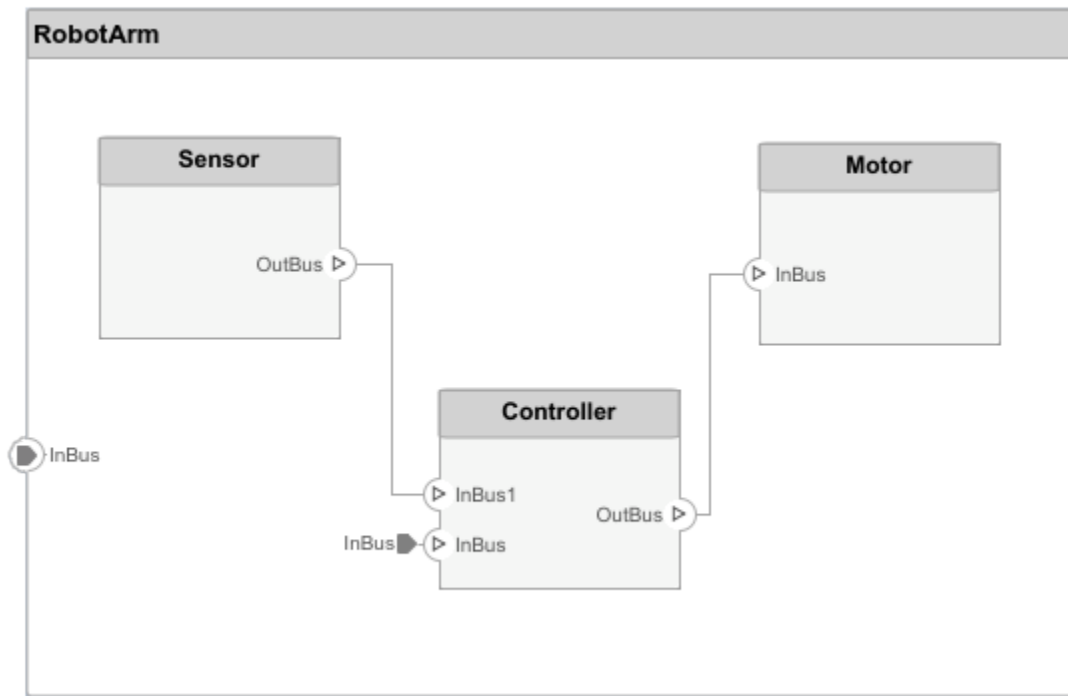
Requirements are integral to the system engineering process. Some are related to the functionality of the overall system, and some are nonfunctional. Some requirements of the overall robot system could be:

- Total power consumption
- Time between issuing a command from the computer and completing the motion of the robot arm
- Precision in positioning the arm

To allocate and trace requirements with system elements, System Composer fully integrates with Simulink® Requirements™

Often, system requirements are broken down into requirements for each component. Component requirements evolve during design.

Robot requirements — maximum speed and sensitivity of proximity sensors — point to subcomponents motors and sensors. You can represent these subcomponents by decomposing the component.



You can associate components with requirements at any level of the system.

Sometimes an overall analysis of the system is necessary to verify requirements or to serve as requirements for the design of other systems. An example would be a box to house the robot system in extreme conditions. Customizing model elements using nonfunctional properties such as weight or temperature sensitivity enable such analyses.

The next step in system design is designing the actual behavior of the components in Simulink. Link System Composer components to Simulink models to trace architectural design to behavioral design.

## See Also

### More About

- “Create an Architecture Model” on page 1-6



- “Inspect Components in Custom Views” on page 1-23
- “Analyze an Architecture Model” on page 1-26

## Create an Architecture Model

### In this section...

“Visually Represent the System” on page 1-6

“Edit Interfaces” on page 1-14

“Decompose Components” on page 1-17

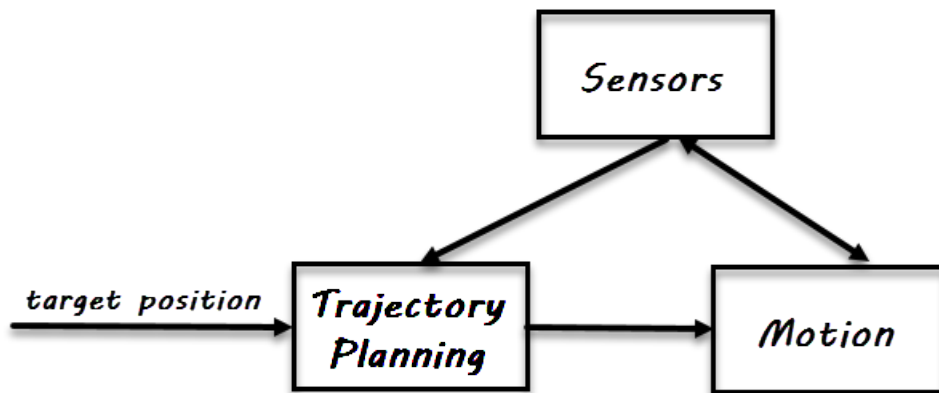
“Implement Component Behavior” on page 1-19

“Link to an Existing Simulink Behavior Model” on page 1-22

In this example, create an architecture model of a mobile robot that consists of sensors, motion, and a planning algorithm. Define the interfaces and link the requirements.

### Visually Represent the System

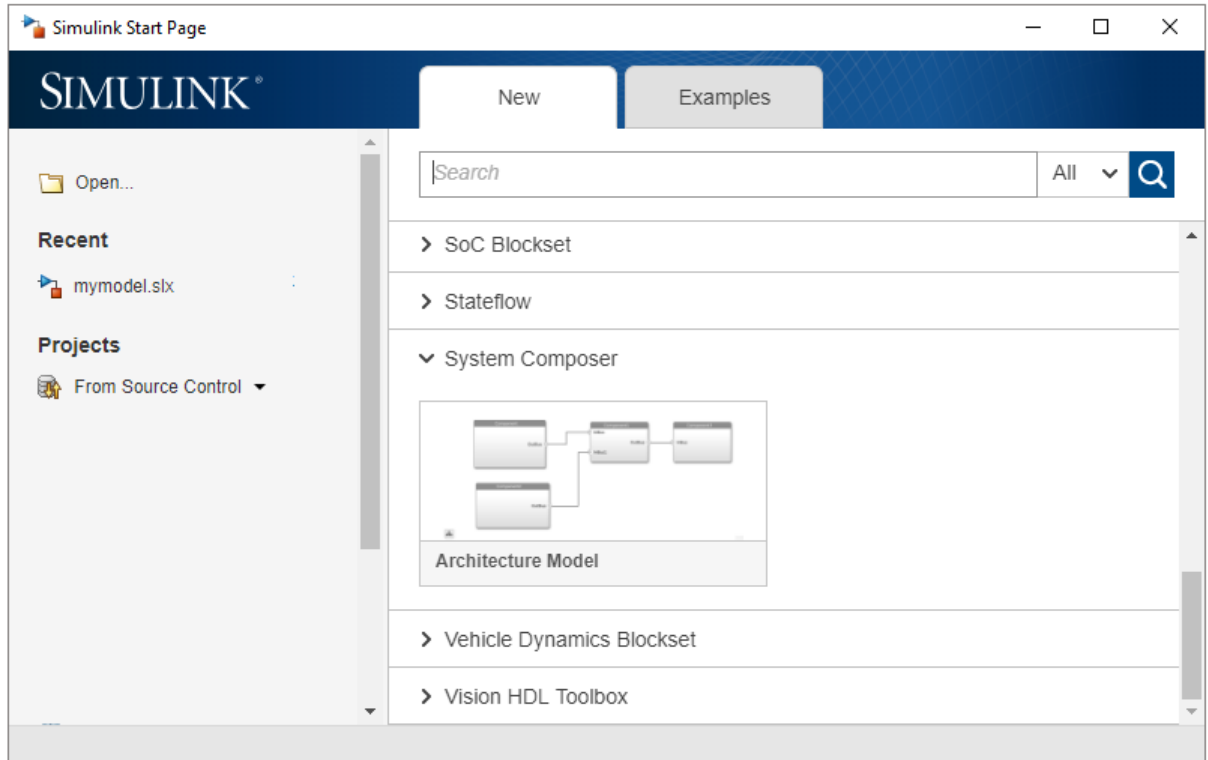
Capture the construction of a robot arm using System Composer. The robot arm consists of these components.



### Create Architecture Model

- 1 In the MATLAB® Command Window, type  
systemcomposer

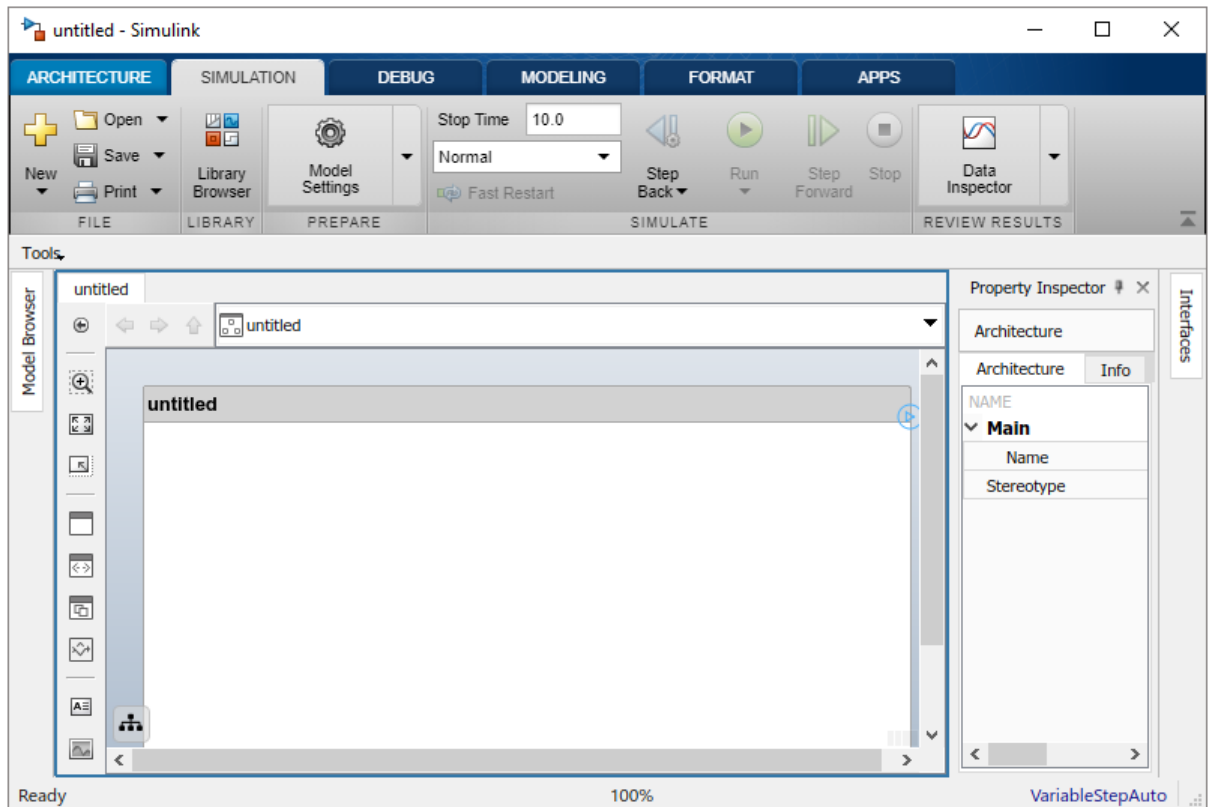
The Simulink Start Page opens to System Composer.



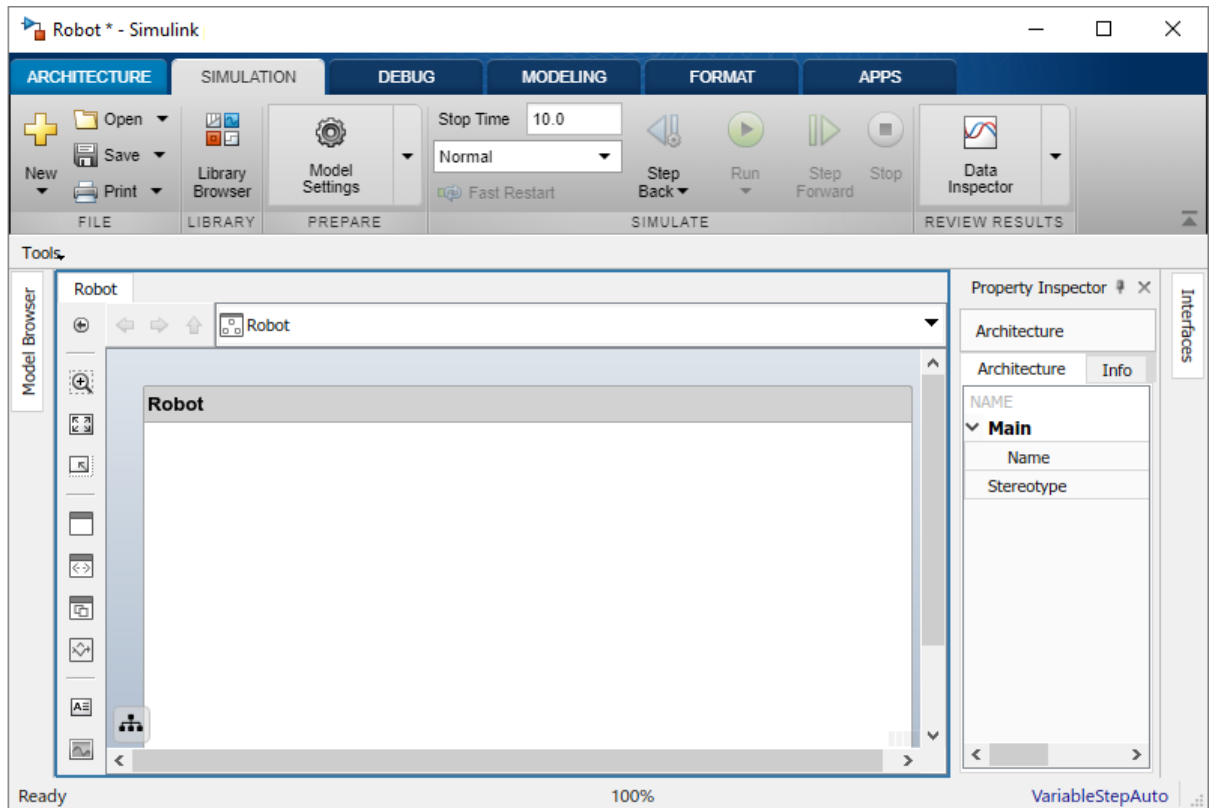
**2 Click Architecture Model.**

A Simulink Editor window opens with a new architecture model. You can identify an architecture model by the badge on the lower left corner and the component palette on the left side.

# 1 Compose an Architecture Model



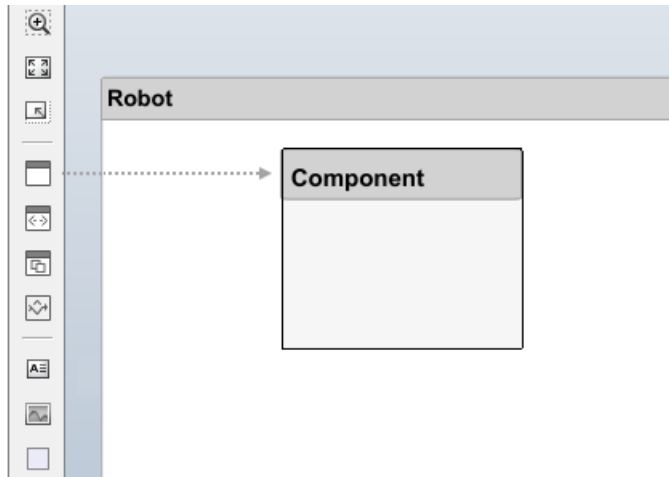
- 3 On the upper-left of the architecture model, double-click `untitled` and name it `Robot`. The name of the model reflects the name of the architecture model.



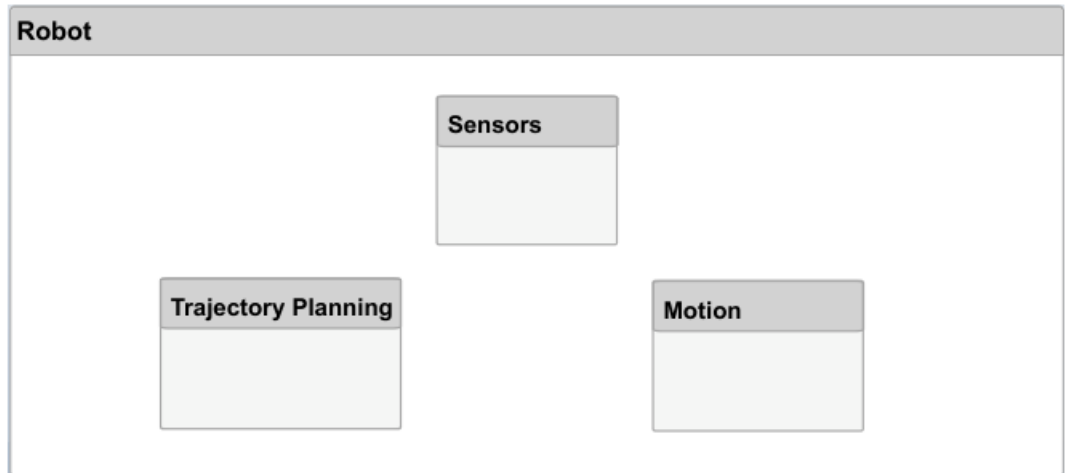
- 4 Save the model.

### Draw Components

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






- 2 Rename the component to Sensors.
- 3 Continue with adding Motion and Trajectory Planning components.

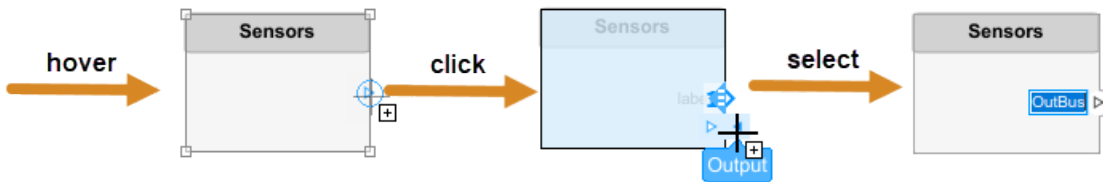


## Create Ports and Connections

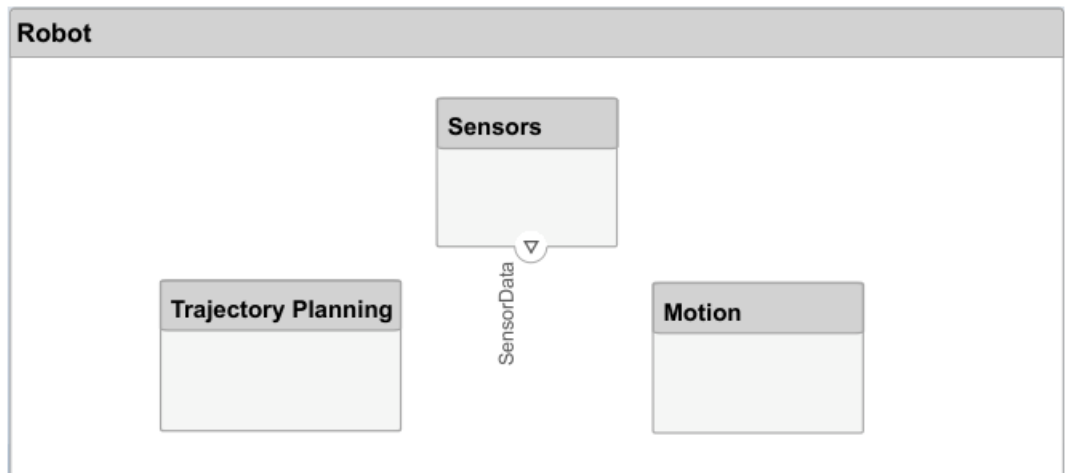
You can add a port to a component on any side and in any direction. To create a port,

pause the mouse cursor over a component side , Click-and-release the mouse button

to view direction options. Select either the input  or output  direction arrow. Rename the port using a proper MATLAB variable name.



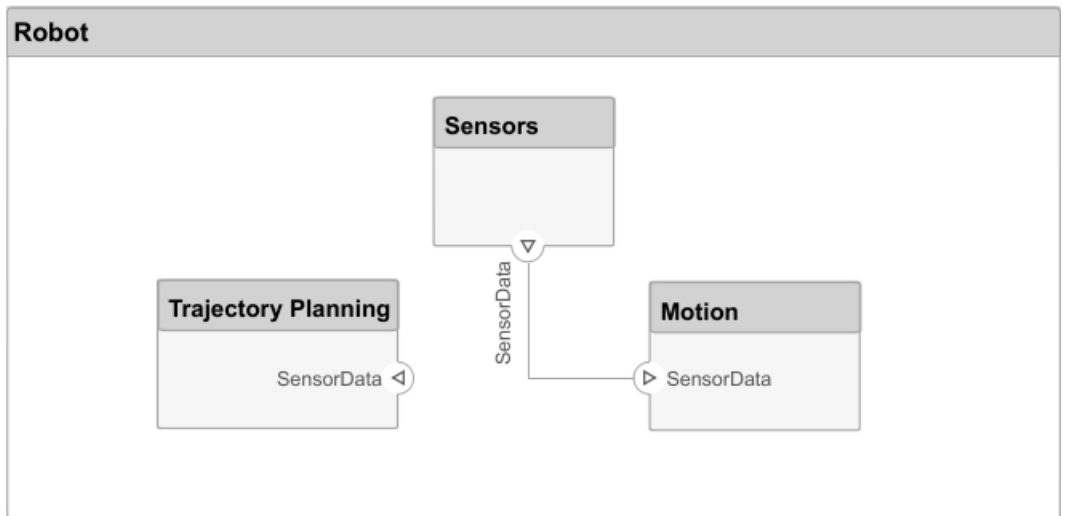
- 1 Create an output port on the bottom side of the Sensors component. Rename it SensorData.



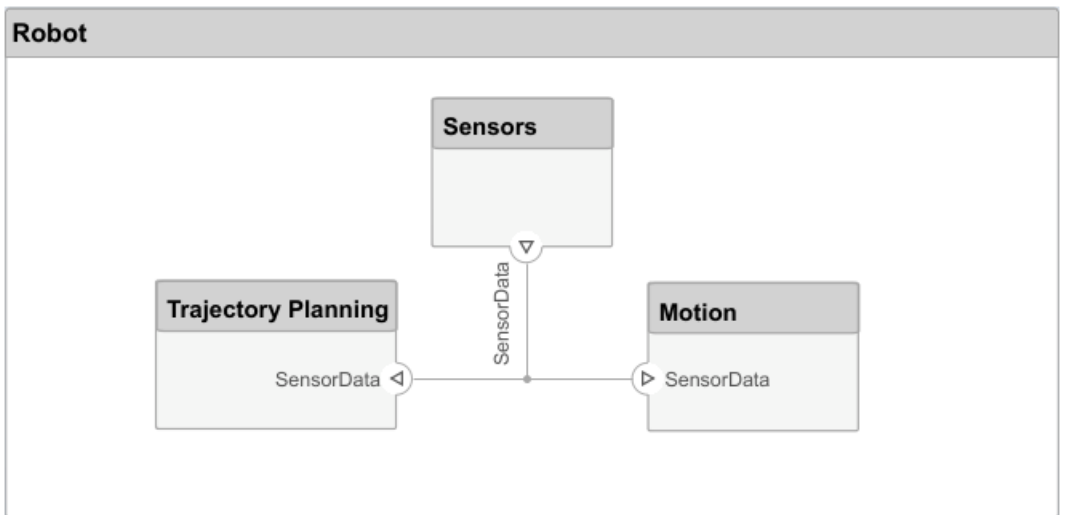
- 2 Click and drag a line from the **SensorData** output port to the Motion component. When you see an input port created at the component side, release the mouse button. By default, this new port has the same name as the source port.

# 1 Compose an Architecture Model

---

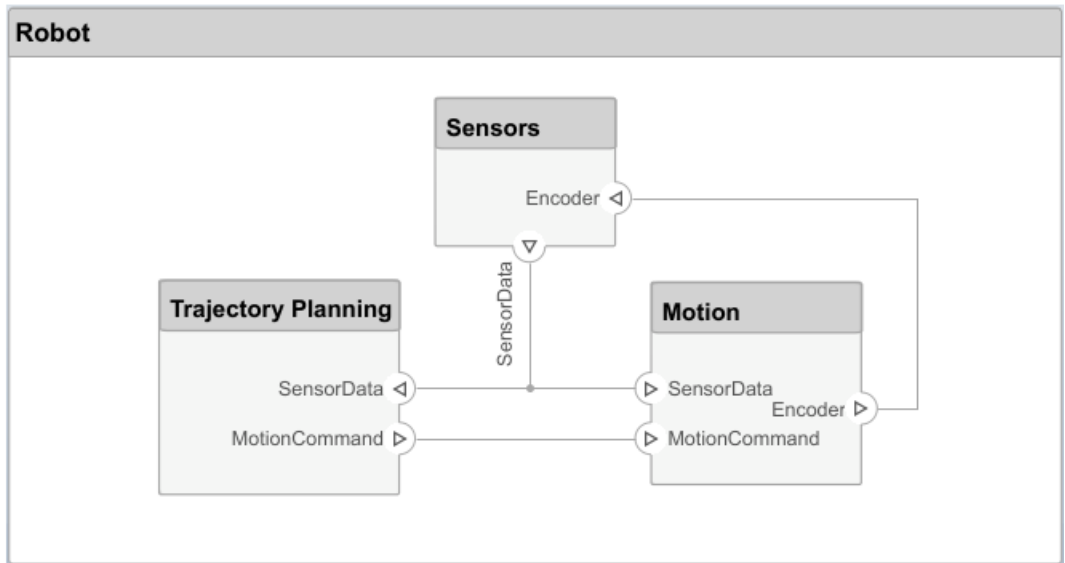


- 3 Hover over the corner of the **SensorData** line until you see the branch icon. Right-click and drag a branch line to the Trajectory Planning component.



- 4 Complete connections as shown in the following figure.



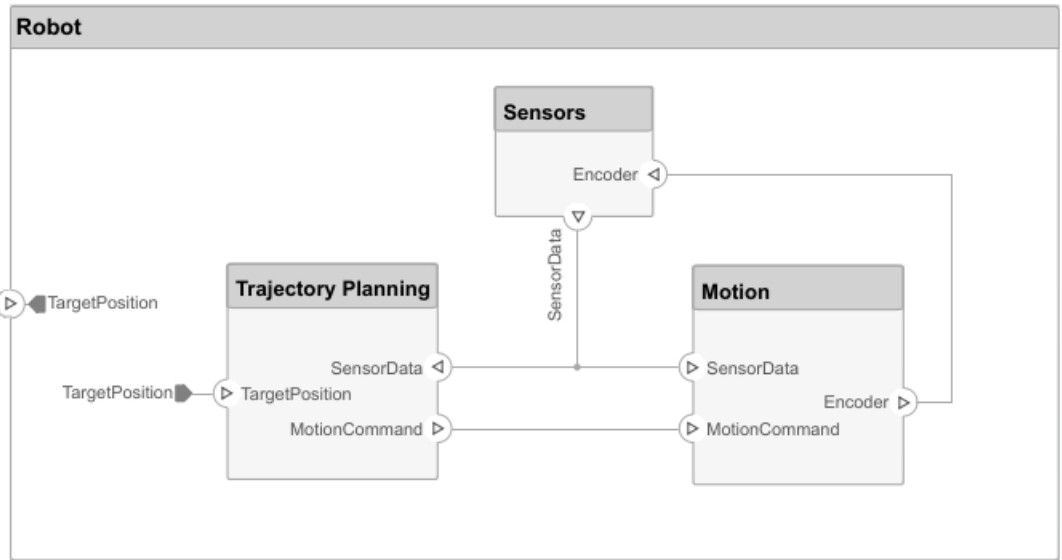


The architecture model can also have ports. In this example, the target position for the robot is provided by a computer external to the robot architecture. Represent this relationship with an input port.

- 1 Click the left edge of the architecture model and type 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 Interfaces

Specify the data flow between components by structuring the data interface with data types, units, dimensions, and so on. An 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 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 it adds to this data set

The data has these specifications.

Name	Data Type	Unit
timestamp	double	seconds
position1 for motor 1	double	degrees

Name	Data Type	Unit
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

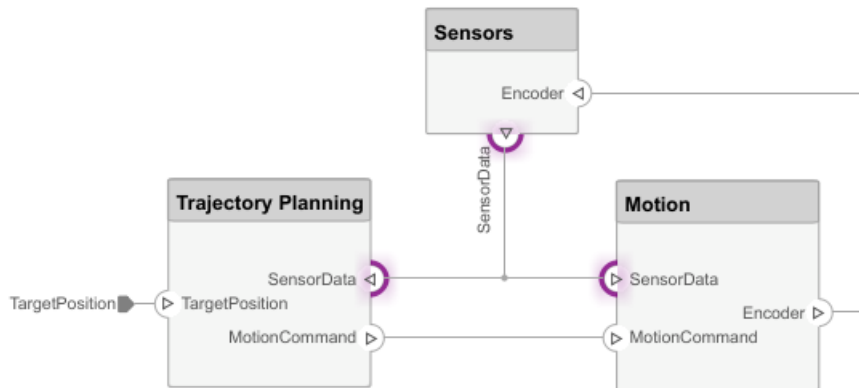
1 On the **Modeling** tab and from the **Design** section, select **Interface Editor** . The interface Editor opens in the right-side pane under the Property Inspector.


2 Click the **Add an interface** button  and name the interface **sensordata**.

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

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

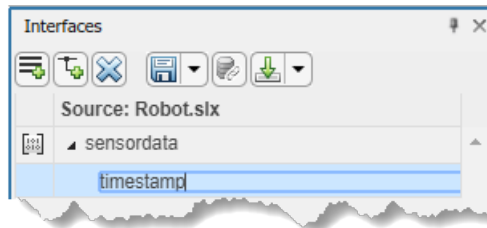
The three **SensorData** ports are highlighted, indicating the ports are associated with the interface **sensordata**.




4 Add an element to the selected interface. Click the add element button  and type **timestamp**.


# 1 Compose an Architecture Model

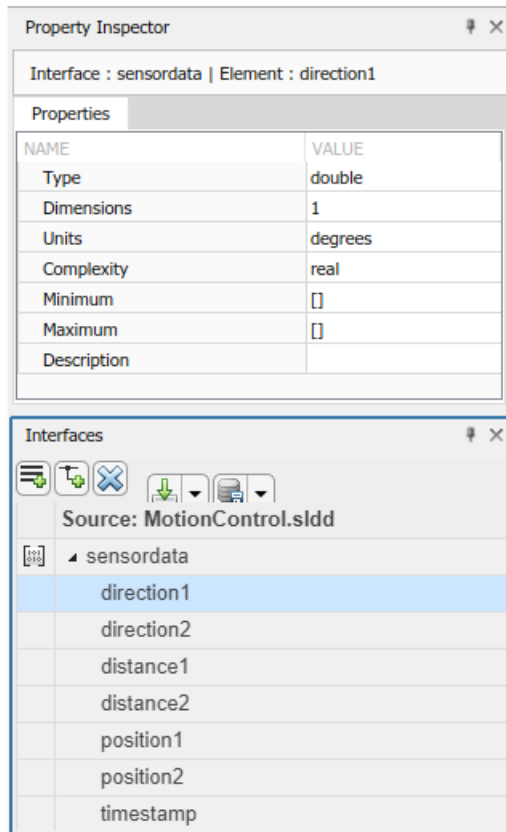
---



- 5 Continue adding elements to the interface by clicking the add element button .
- 6 Edit the properties of an interface element using the **Property Inspector**. Right-click on any element and select **Inspect Properties**. The Property Inspector opens above the Interfaces editor.

Click each interface element and add units as shown in the specification. Click the

drop-down next to the  button to save the interface to a data dictionary. Here it is saved as **MotionControl**.



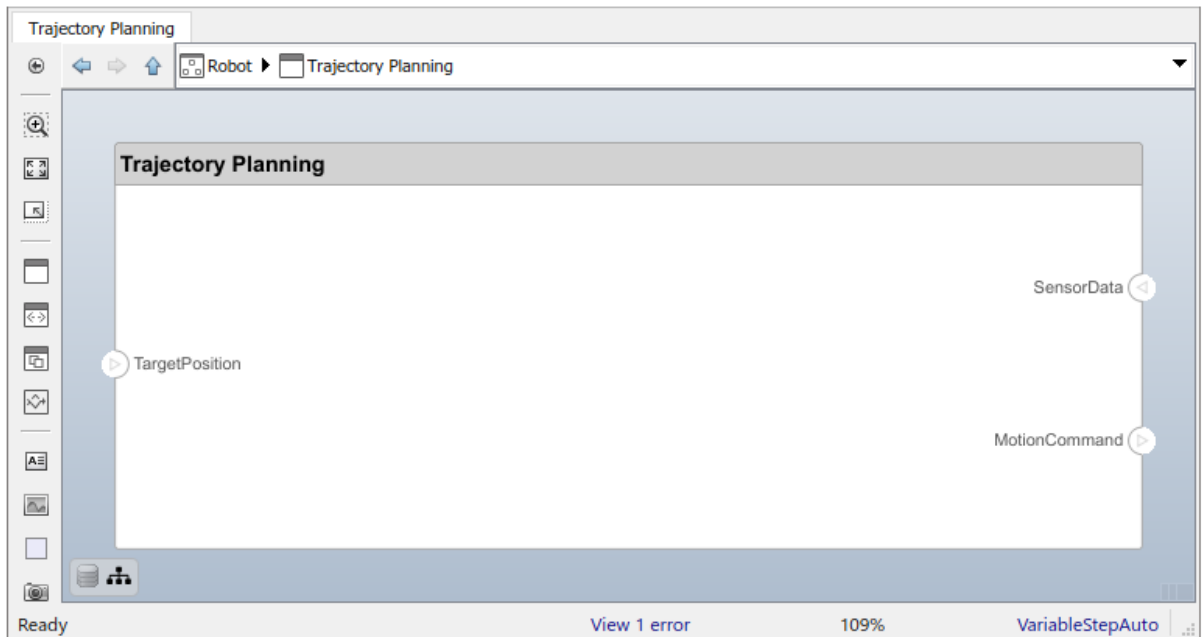
## Decompose Components

Each component can have an architecture of its own. 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.

## 1 Compose an Architecture Model

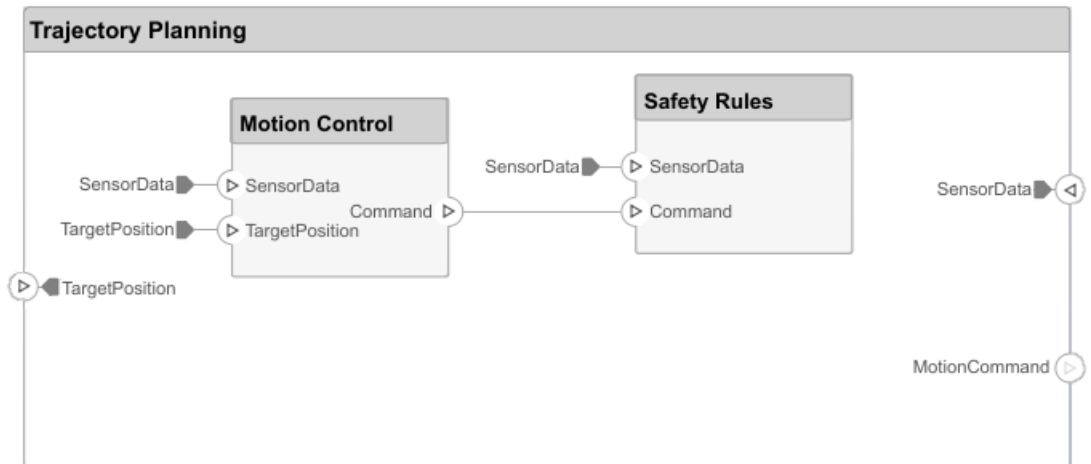
---



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



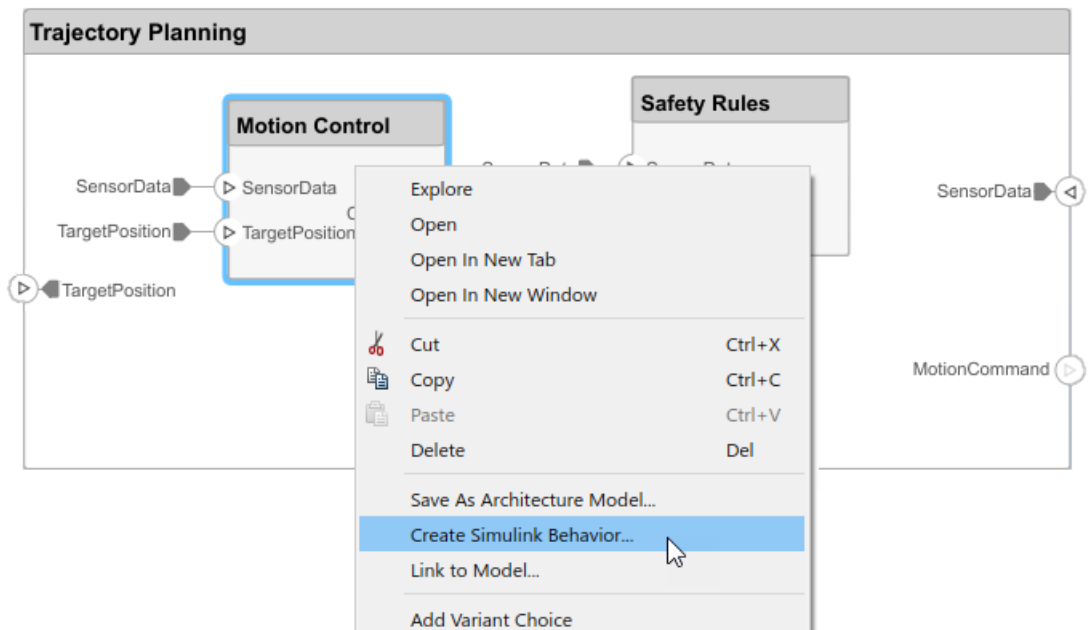
## Implement Component Behavior

If you have a Component block that represents a single functional unit that does not need further architectural decomposition, you can define its behavior in Simulink.

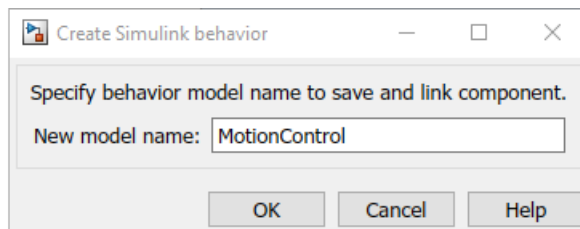
- 1 Right-click the Motion Control block and select **Create Simulink Behavior**.

## 1 Compose an Architecture Model

---

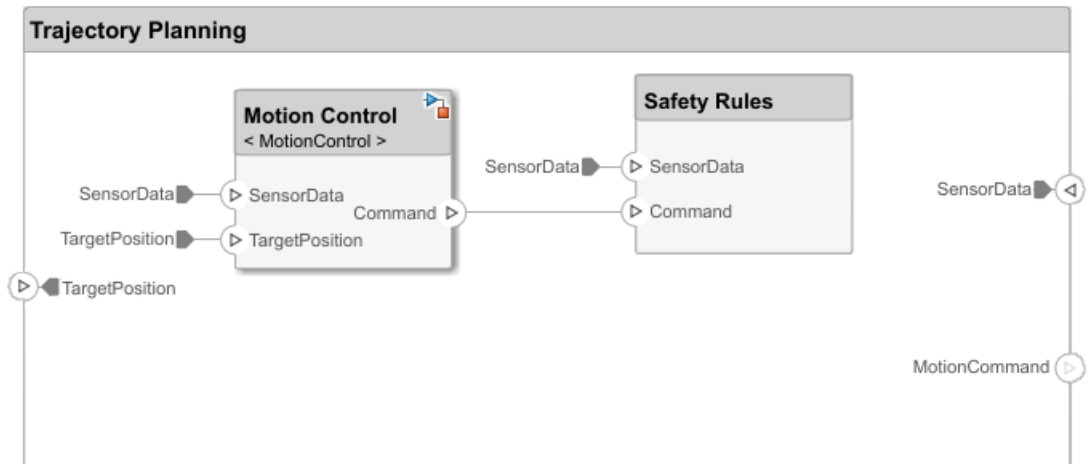


- 2 Type the name for a new Simulink model and click **OK**.

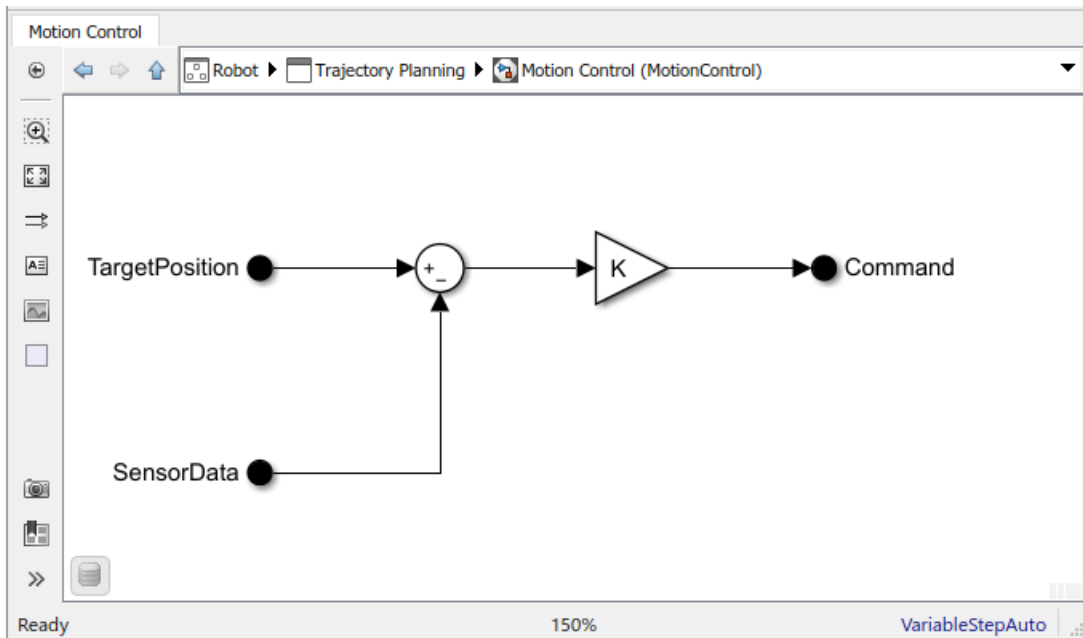


This creates a new Simulink model in the current directory, converts the Component block to a Reference Component block, and indicates a link by adding a Simulink icon with the name of the referenced model to the block title. The Simulink behavior model reflects the ports from the component in the architecture model and allows you to define the behavior in Simulink.





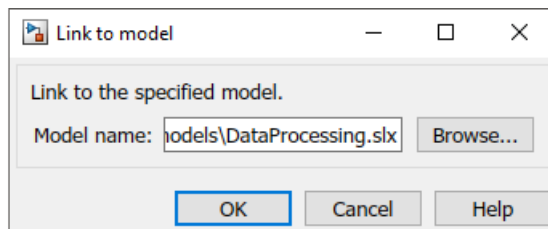
- 3 Double-click Motion Control block to open the Simulink Editor window.
- 4 Add a Sum block to subtract SensorData from TargetPosition, and add a Gain block before connecting to the output Command.



## Link to an Existing Simulink Behavior Model

You can link to an existing Simulink behavior model from a System Composer component.

- 1 Right-click the component and select **Link to Model**.
- 2 Enter the name of a Simulink model.



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

## See Also

### More About

- “Analyze an Architecture Model” on page 1-26
- “Create Spotlight Views”

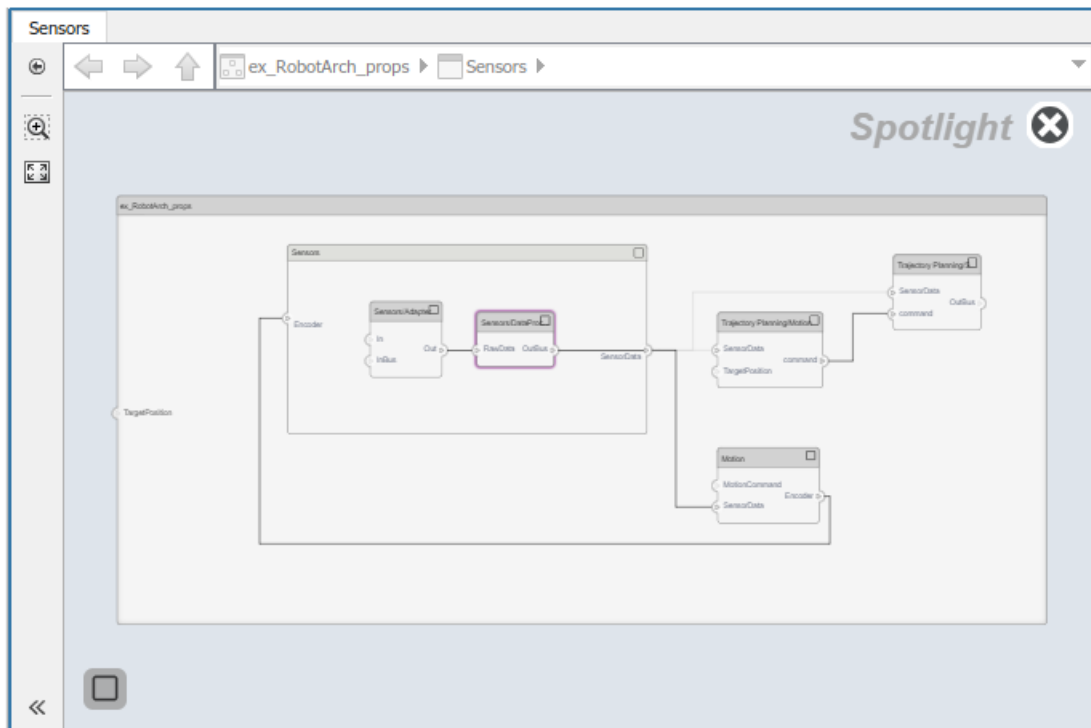
## Inspect Components in Custom Views

View the hierarchy and connectivity of a component in a specialized view.

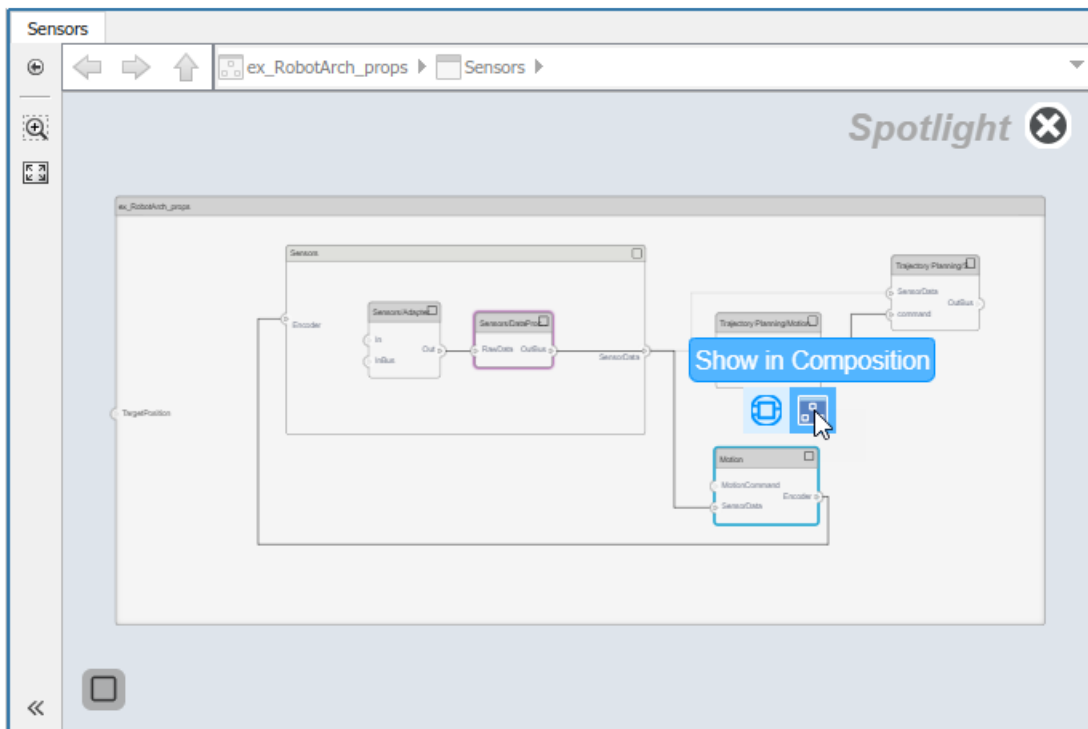
- 1 Open the architecture model for a robot, `ex_RobotArch_props` using the MATLAB command  


```
open_system('ex_RobotArch_props')
```
- 2 Double-click the Sensors component, and then select the DataProcessing component.
- 3 Right-click and select **Create Spotlight From Component**.

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. Spotlight views are transient; they are not saved with the model.







- 6 Return to the architecture model view by clicking the  icon.

## See Also

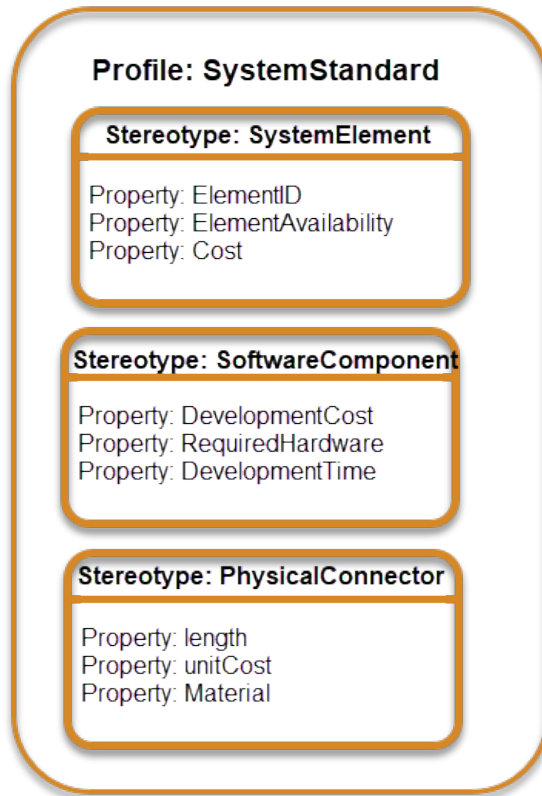
### More About

- “Analyze an Architecture Model” on page 1-26

## Analyze an Architecture Model

In this section...
“Load Architecture Model Profile” on page 1-27
“Apply Stereotypes to Model Elements” on page 1-29
“Set Properties” on page 1-32
“Perform an Analysis” on page 1-35

A profile contains a set of model element stereotypes with custom properties. A stereotype can be applicable to components, ports, connections, and architectures, or it can be applicable to a specific element type, such as components. When a model element has a stereotype, you can specify property values as part of its architectural definition. Stereotypes and properties also make analysis of an architecture possible.



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

The goal of this example is to compute the total cost of the system given the cost of its parts. The example profile is limited to this goal.

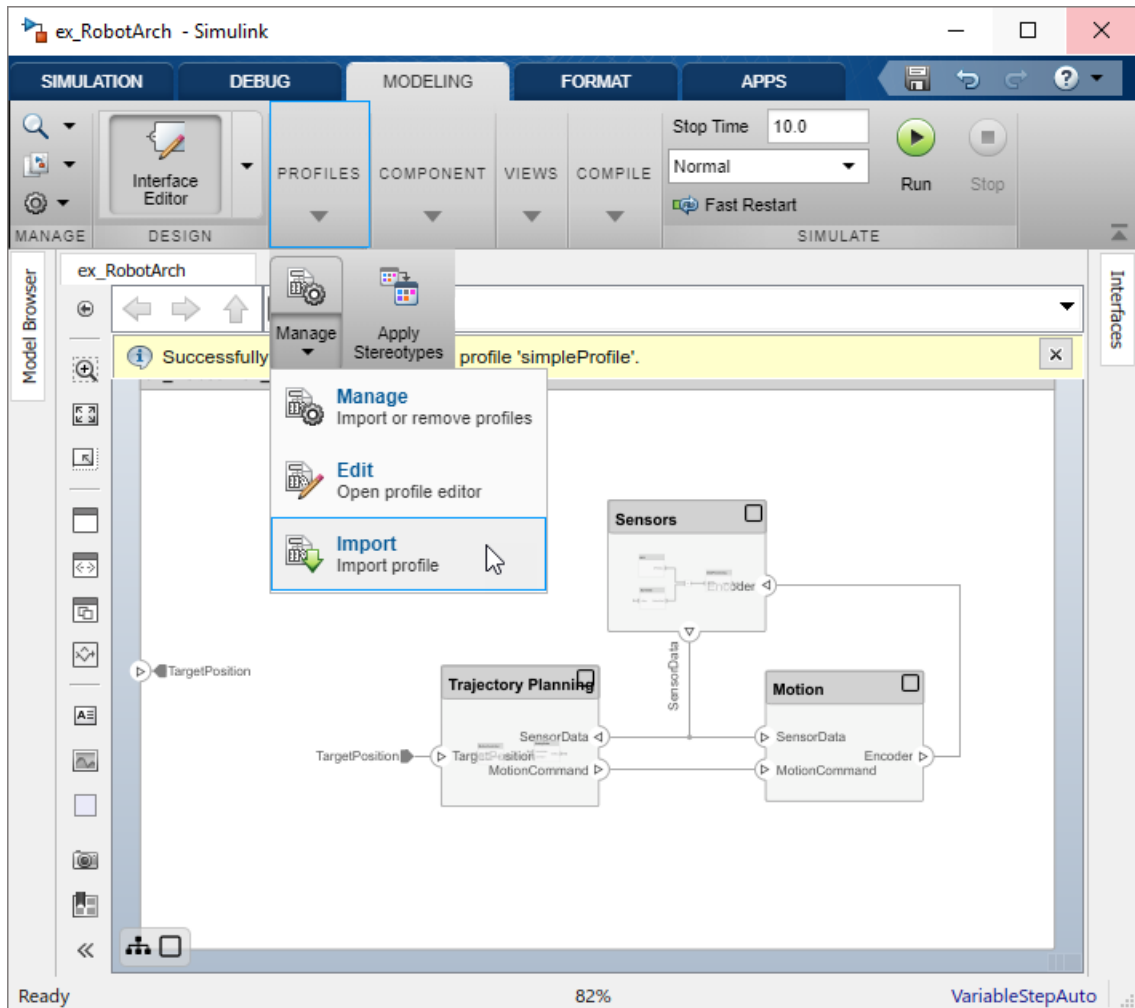
## Load Architecture Model Profile

Load a profile to make stereotypes available for model elements.

- 1 On the **Model** tab and in the **Profiles** section, select **Manage** and then from the drop-down, select **Import** .

# 1 Compose an Architecture Model

- 2 Browse to the examples folder. <matlabroot>\toolbox\systemcomposer\examples.
- 3 Select simpleProfile.xml.



This profile contains 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.

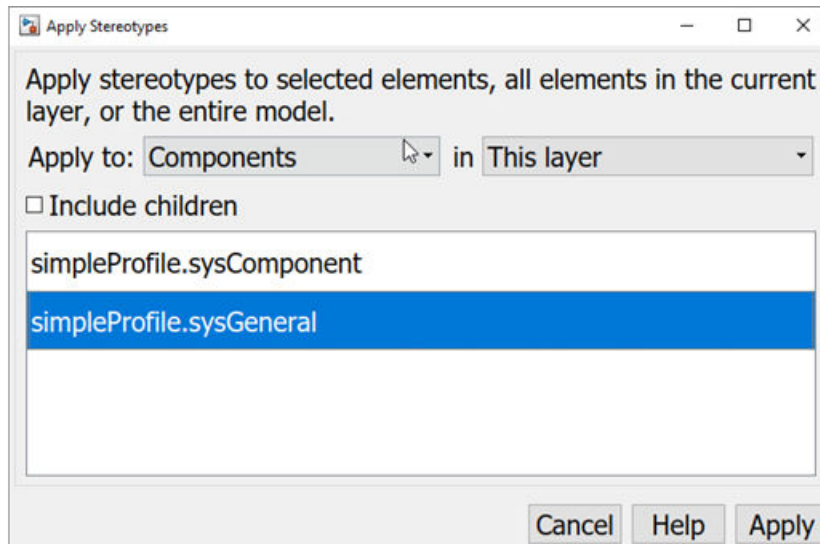
- `sysGeneral` is a general stereotype, applicable to all element types, that enables adding generic properties such as an ID that helps identify the element throughout the design and implementation process, and a Note that project members can use to track any issues with the element.
- `sysComponent` 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 price and weight properties defined per meter of length. These properties help compute the total weight and cost of the design in this particular example.

## Apply Stereotypes to Model Elements

Add custom properties to a model element by applying a stereotype from a loaded profile. This procedure uses the model `ex_RobotArch.slx`.

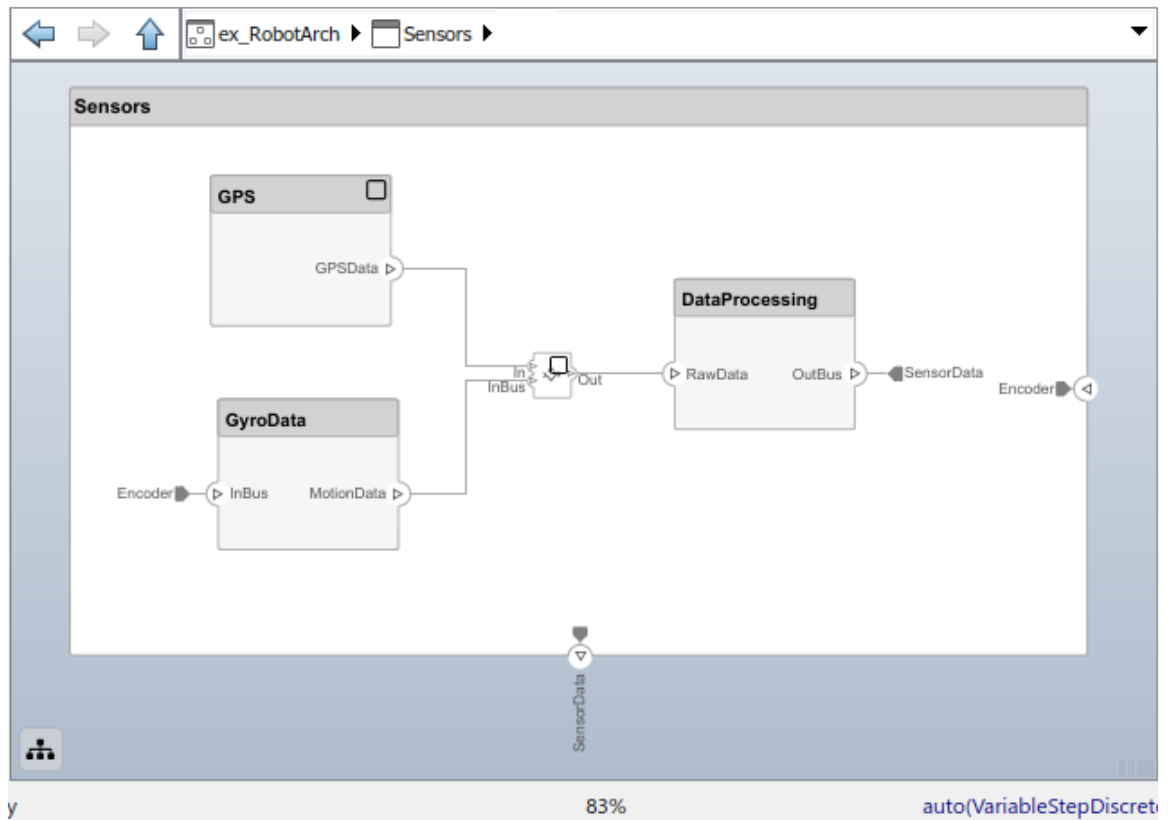
- 1 Open the example robot model with the MATLAB command `open_system('ex_RobotArch_props.slx')`.
- 2 Open the Sensors component.
- 3 On the **Modeling** tab and in the **Profiles** section, select **Apply Stereotypes**.
- 4 In the Apply Stereotypes dialog box and from the **Apply to** list, select **All elements**. From the **in** list, select **This layer**.

In the list of available stereotypes, select `simpleProfile.sysGeneral`. The stereotype `sysGeneral` is useful for tracking.



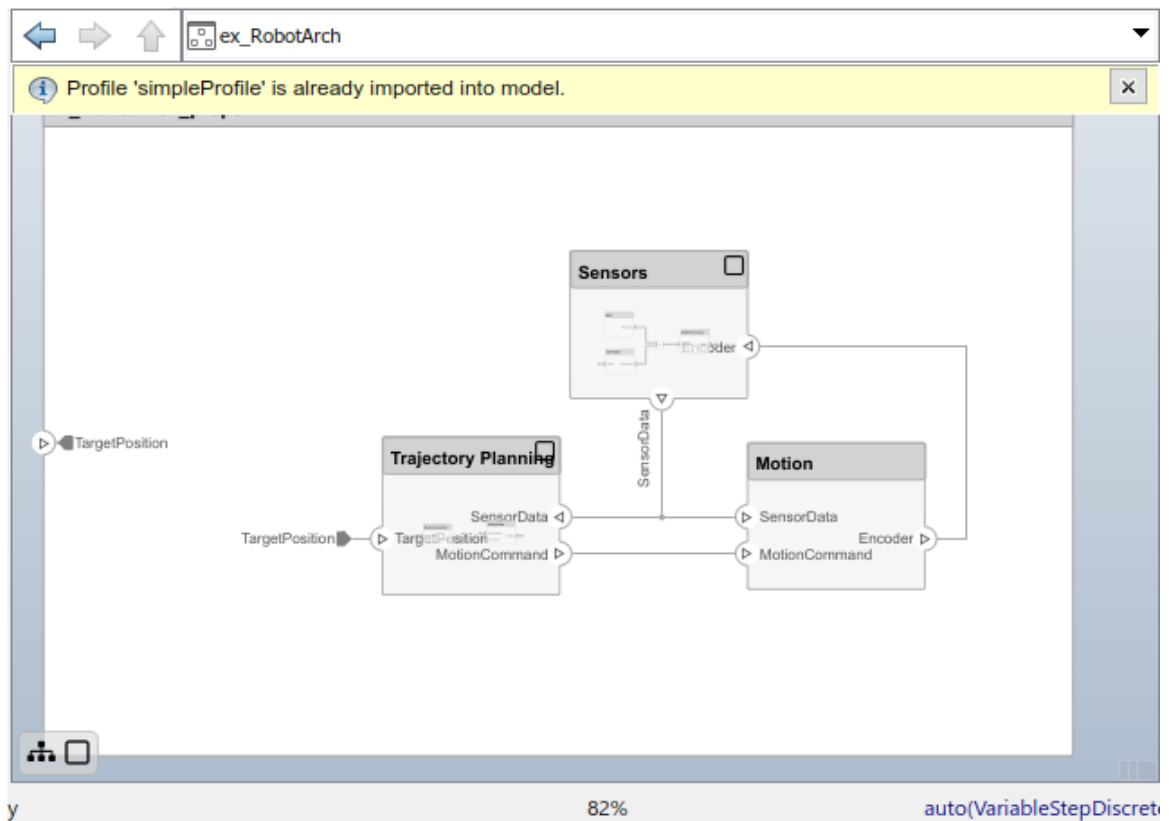
- 5 Click **Apply**.
- 6 In the Sensors component, select the GPS component. Right-click and select **Apply Stereotype > simpleProfile.sysComponent**.

The stereotype `sysComponent` is useful for physical properties and cost. `sysGeneral` was selected in a previous step.



- 7 Navigate to 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, and select **Apply Stereotype > simpleProfile.sysComponent**.
- 8 Apply the `sysConnector` stereotype to all connectors in the Sensors layer, the Trajectory Planning layer and the top model layer.

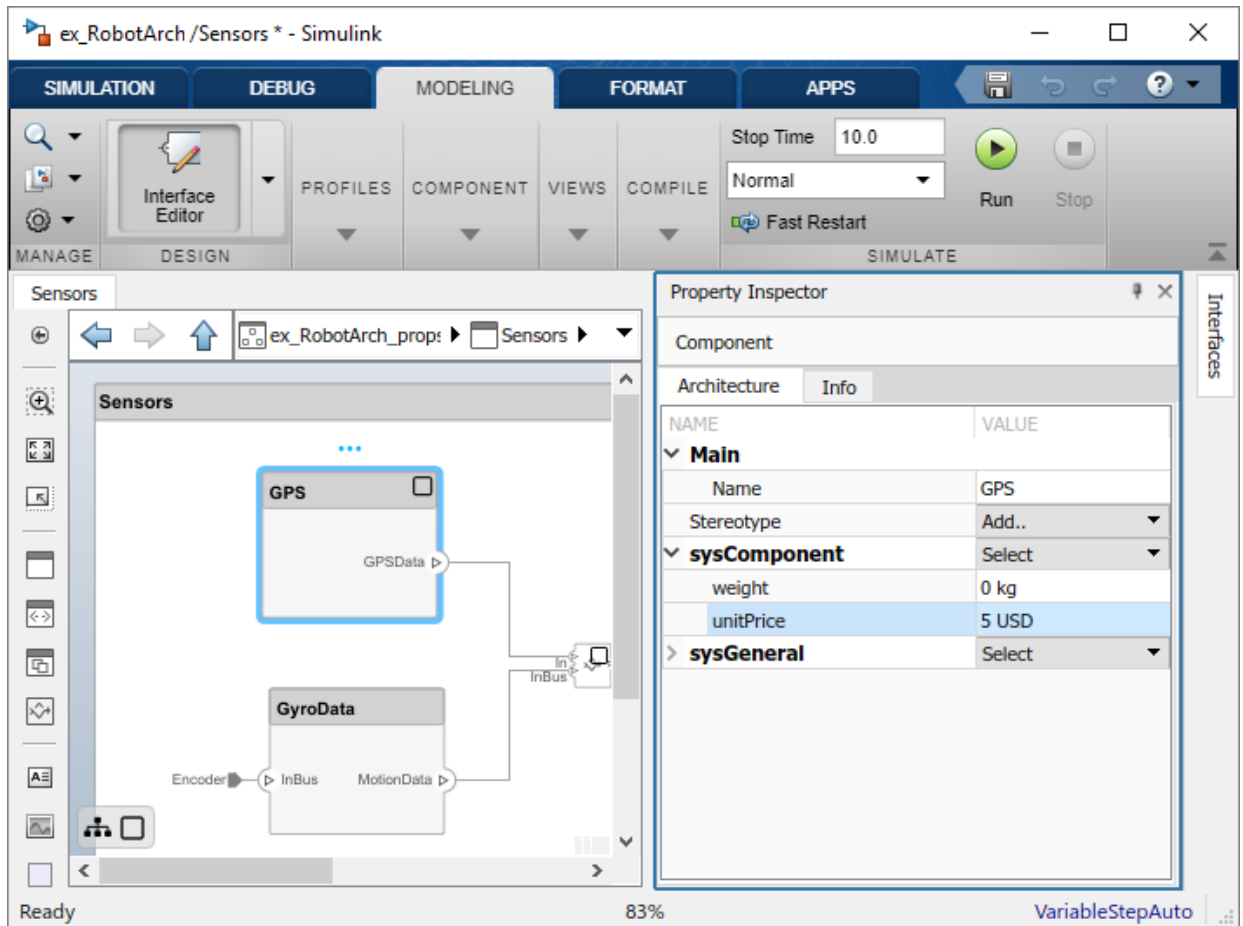
# 1 Compose an Architecture Model



## Set Properties

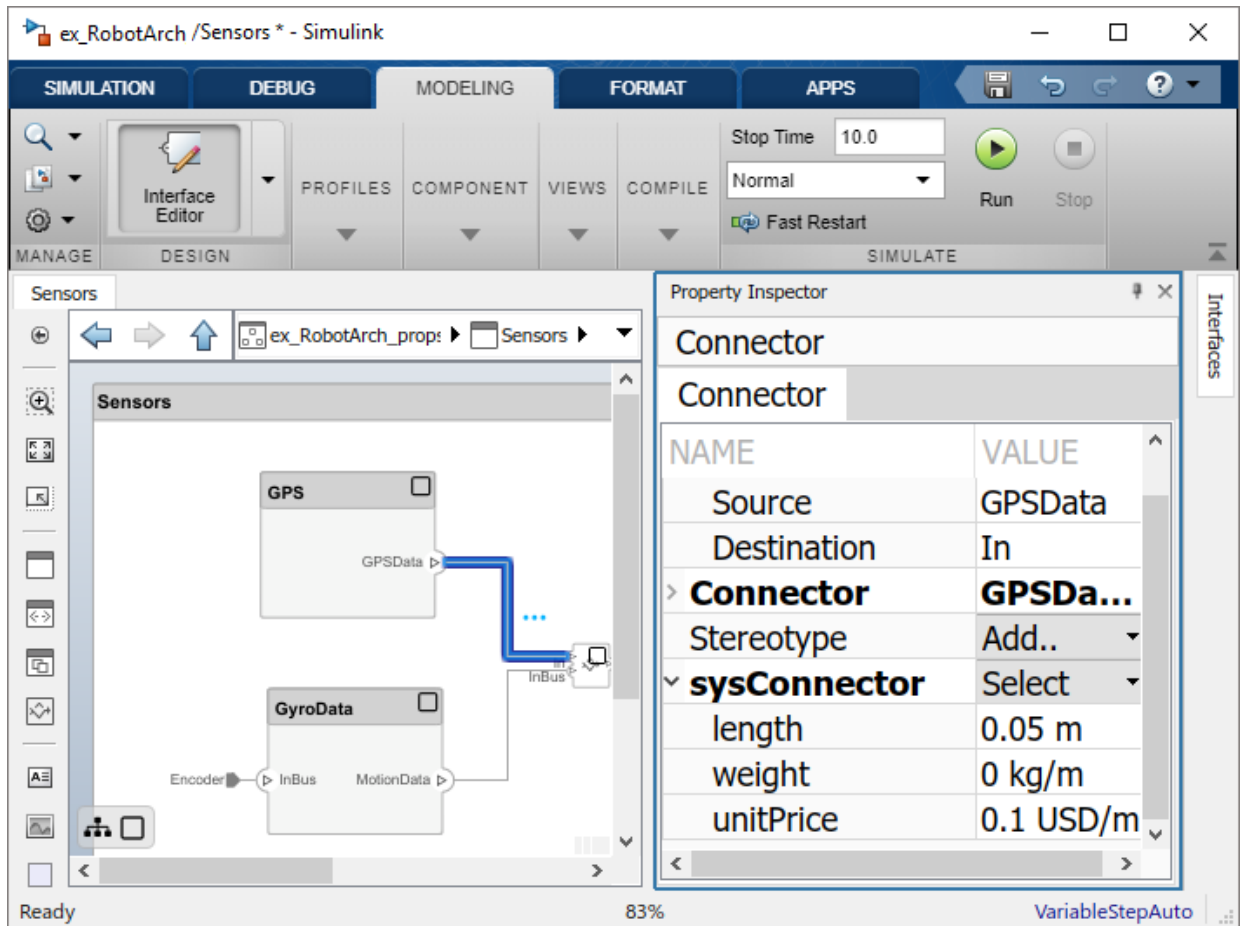
Set the property values to enable cost analysis. Follow this example for the GPS module.

- 1 In the Sensors component, select GPS component.
- 2 Open the Property Inspector. On the **Modeling** tab and in the **Design** section, select **Property Editor**.
- 3 Expand the **sysComponent** stereotype to see the properties.
- 4 Set **unitPrice** to 5 and press Enter.



- 5 Similarly, for the GPSData connector properties, set the **length** to 0.05 and **unitPrice** to 0.1.

# 1 Compose an Architecture Model



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

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

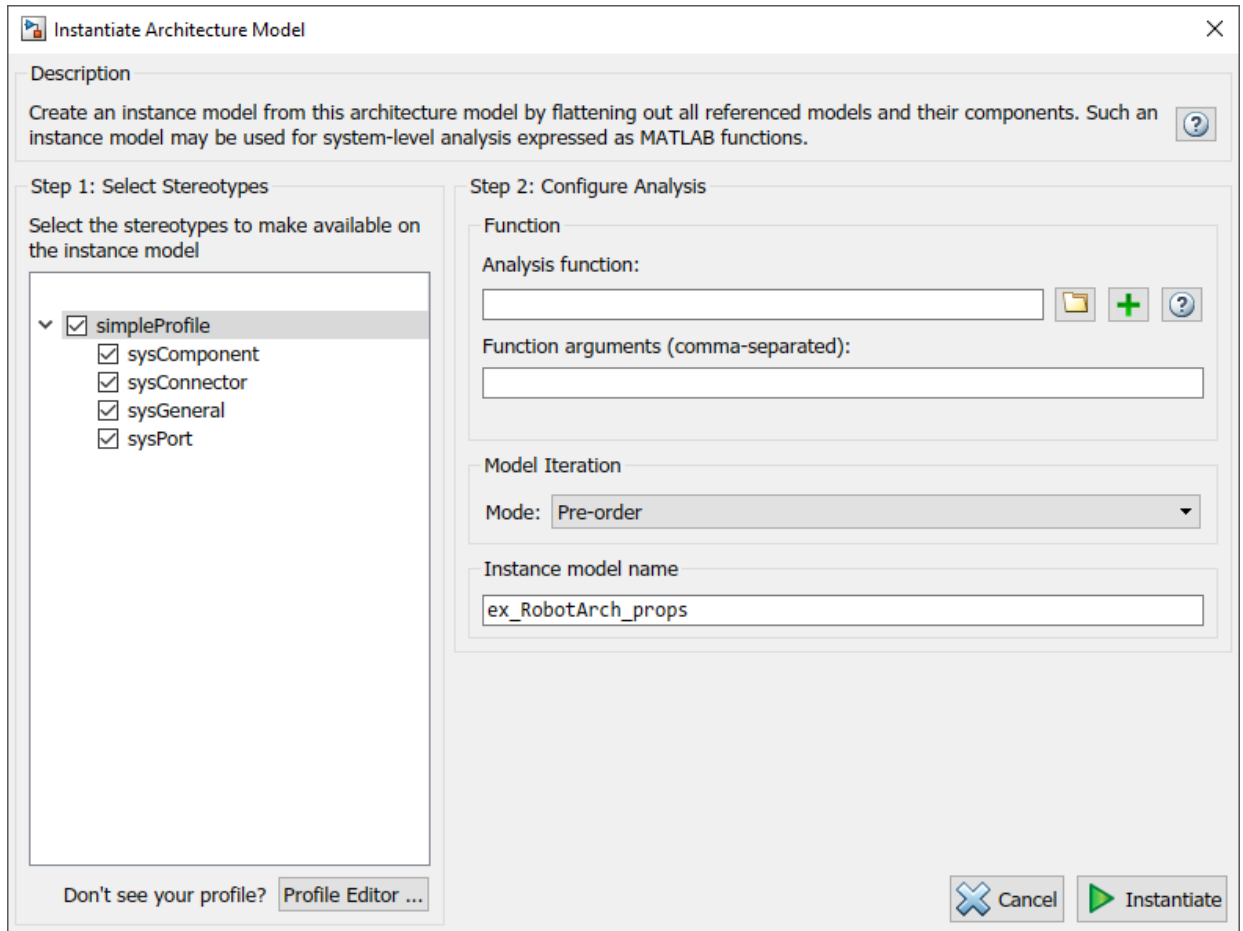
- 7 Save the model as `ex_RobotArch_props.slx`.

## Perform an Analysis


Analyze the total cost for all components in the robot model.

- 1 On the **Modeling** tab and in the **Views** section, select **Analysis Model**, and then from the drop-down list select **Analysis Model**.

# 1 Compose an Architecture Model



- 2 Add an analysis function. In the Analysis function box, enter the function name

ex\_RobotArch\_analysis without an extension, and then click . 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 to obtain total cost for their parent architecture. This function computes the cost for one model element.

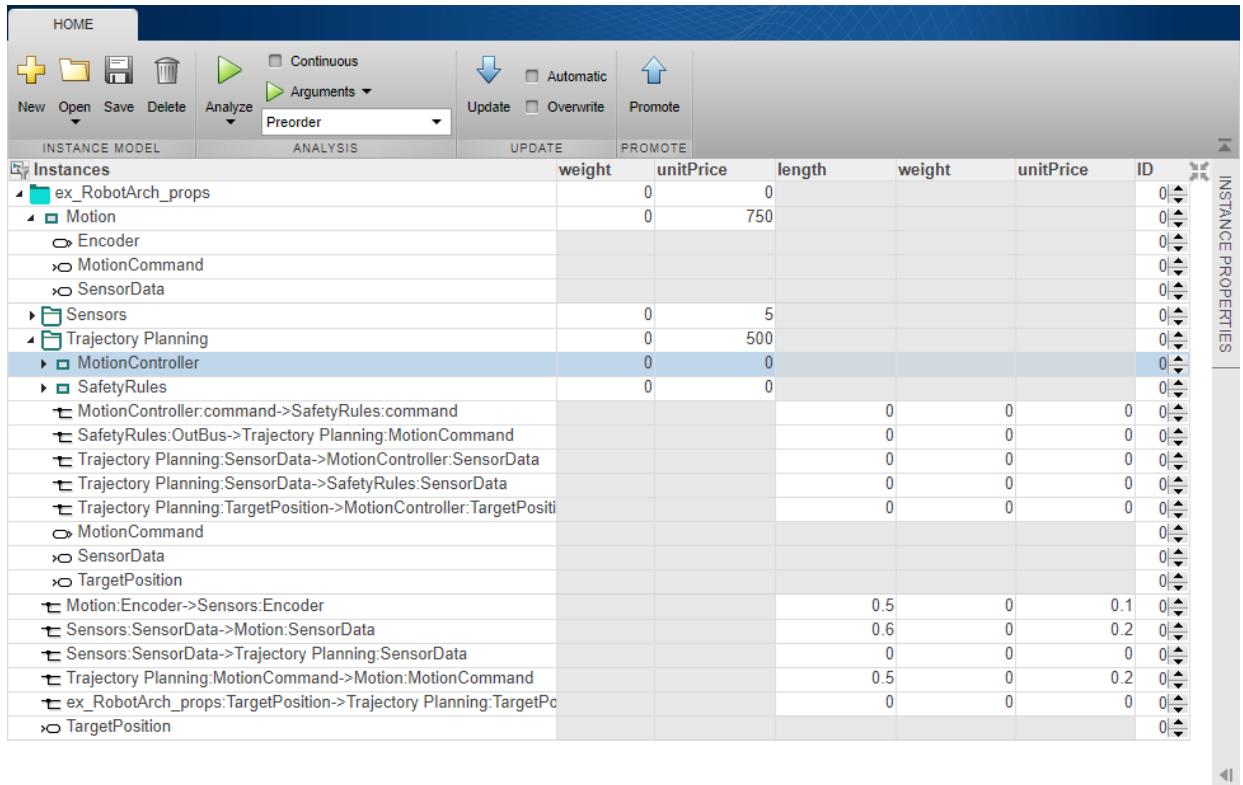


```
function ex_RobotArch_analysis(instance,varargin)

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 + sysComponent_unitPrice;
    end
    instance.setValue("sysComponent.unitPrice",sysComponent_unitPrice)
end
```

- 3 Return to the Analysis screen and click **Instantiate**. The analysis viewer shows the properties of each model element.

# 1 Compose an Architecture Model



Instances	weight	unitPrice	length	weight	unitPrice	ID
ex_RobotArch_props	0	0				0
Motion	0	750				0
Encoder						0
MotionCommand						0
SensorData						0
Sensors	0	5				0
Trajectory Planning	0	500				0
MotionController	0	0				0
SafetyRules	0	0				0
MotionController:command->SafetyRules:command			0	0	0	0
SafetyRules:OutBus->Trajectory Planning:MotionCommand			0	0	0	0
Trajectory Planning:SensorData->MotionController:SensorData			0	0	0	0
Trajectory Planning:SensorData->SafetyRules:SensorData			0	0	0	0
Trajectory Planning:TargetPosition->MotionController:TargetPositi			0	0	0	0
MotionCommand						0
SensorData						0
TargetPosition						0
Motion:Encoder->Sensors:Encoder			0.5	0	0.1	0
Sensors:SensorData->Motion:SensorData			0.6	0	0.2	0
Sensors:SensorData->Trajectory Planning:SensorData			0	0	0	0
Trajectory Planning:MotionCommand->Motion:MotionCommand			0.5	0	0.2	0
ex_RobotArch_props:TargetPosition->Trajectory Planning:TargetPo			0	0	0	0
TargetPosition						0

- 4 Select **Bottomup** as the iteration method and click **Analyze**.

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



## See Also

### More About

- “Create an Architecture Model” on page 1-6

- “Define Profiles and Stereotypes”



# Refactor a Simulink Model as a Composition

---

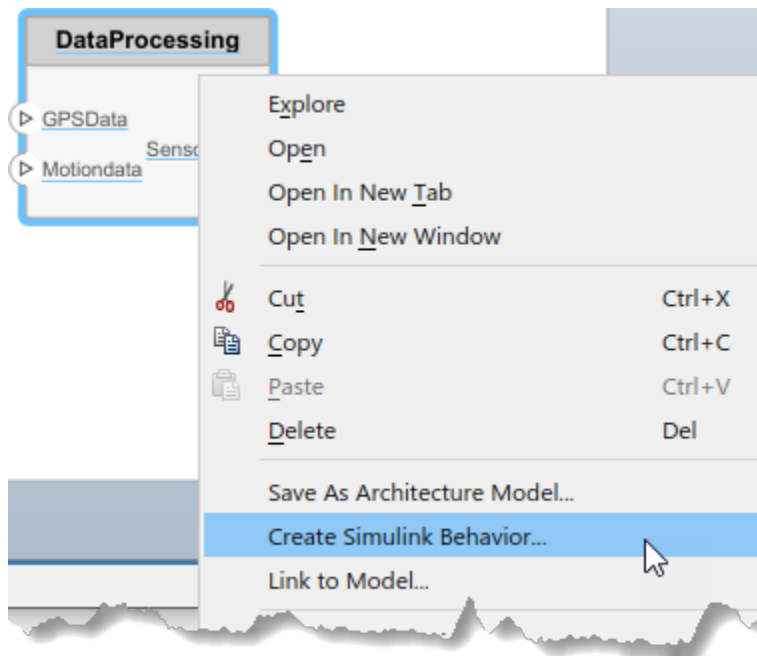
- “Implement Component Behavior in Simulink” on page 2-2
- “Export Simulink Model as Architecture” on page 2-6

# Implement Component Behavior in Simulink

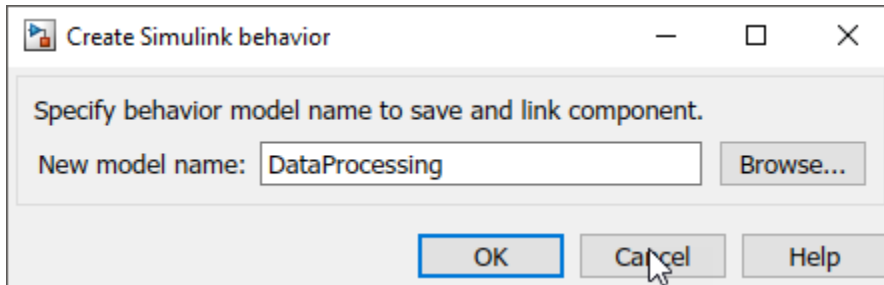
Create and use Simulink models to specify component behavior.

## Create a Simulink Behavior Model

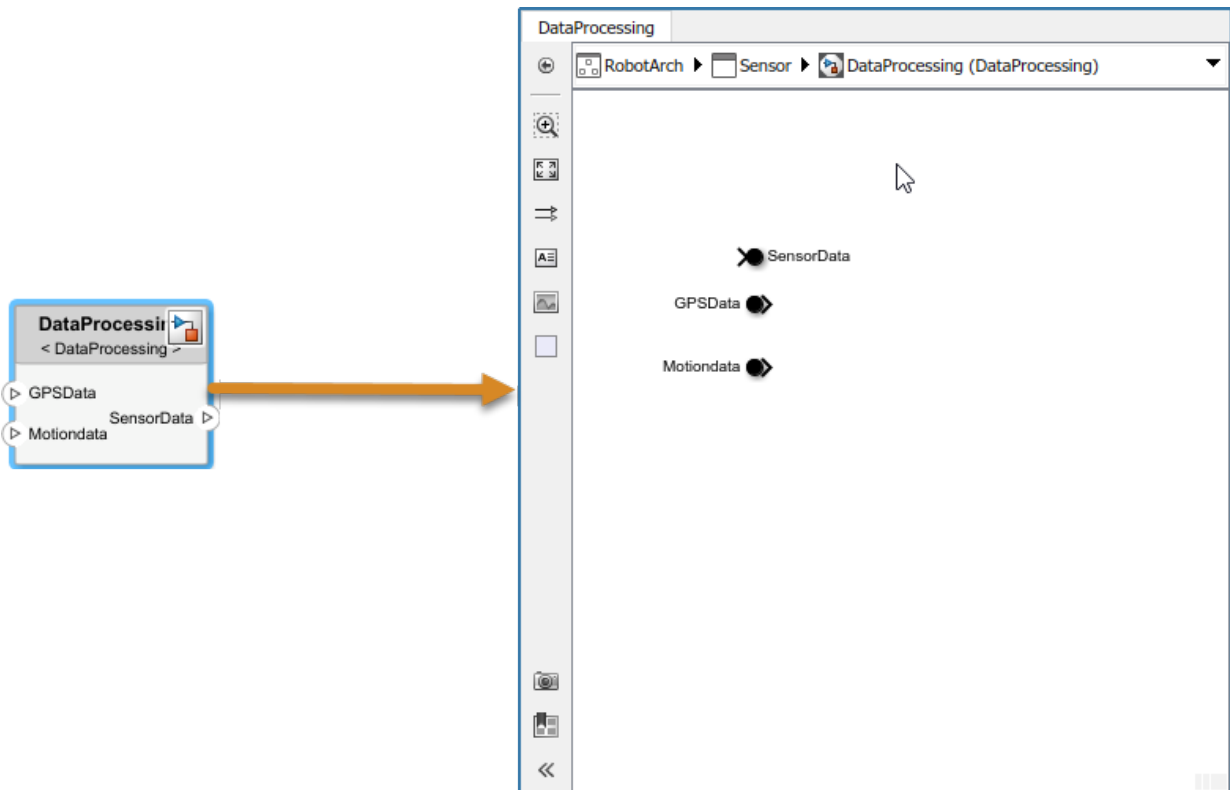
When a component does not require further architectural decomposition, you can design its behavior in Simulink. Right-click the component and select **Create Simulink Behavior**.



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

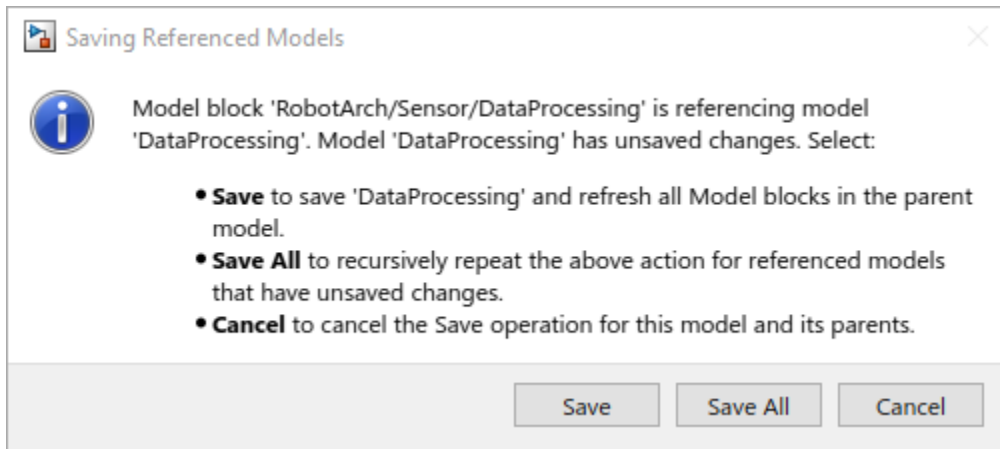


- A new Simulink model with the provided name is created. The root level ports of the Simulink model reflect the ports of the component.
- The component in the architecture model is linked to the Simulink model. The Simulink icon on the component indicates a Simulink link.



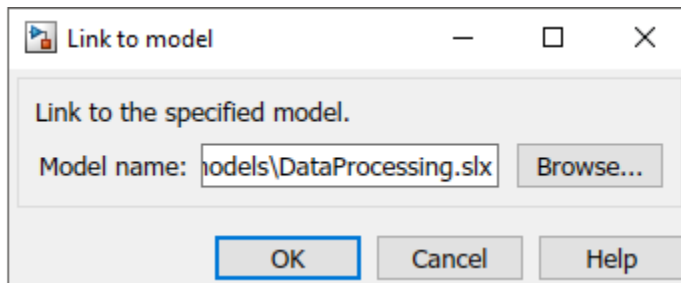
You can now go on to provide specific dynamics and algorithms in the referenced Simulink model. Adding root-level ports in the Simulink model creates additional ports on the referencing component.

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.



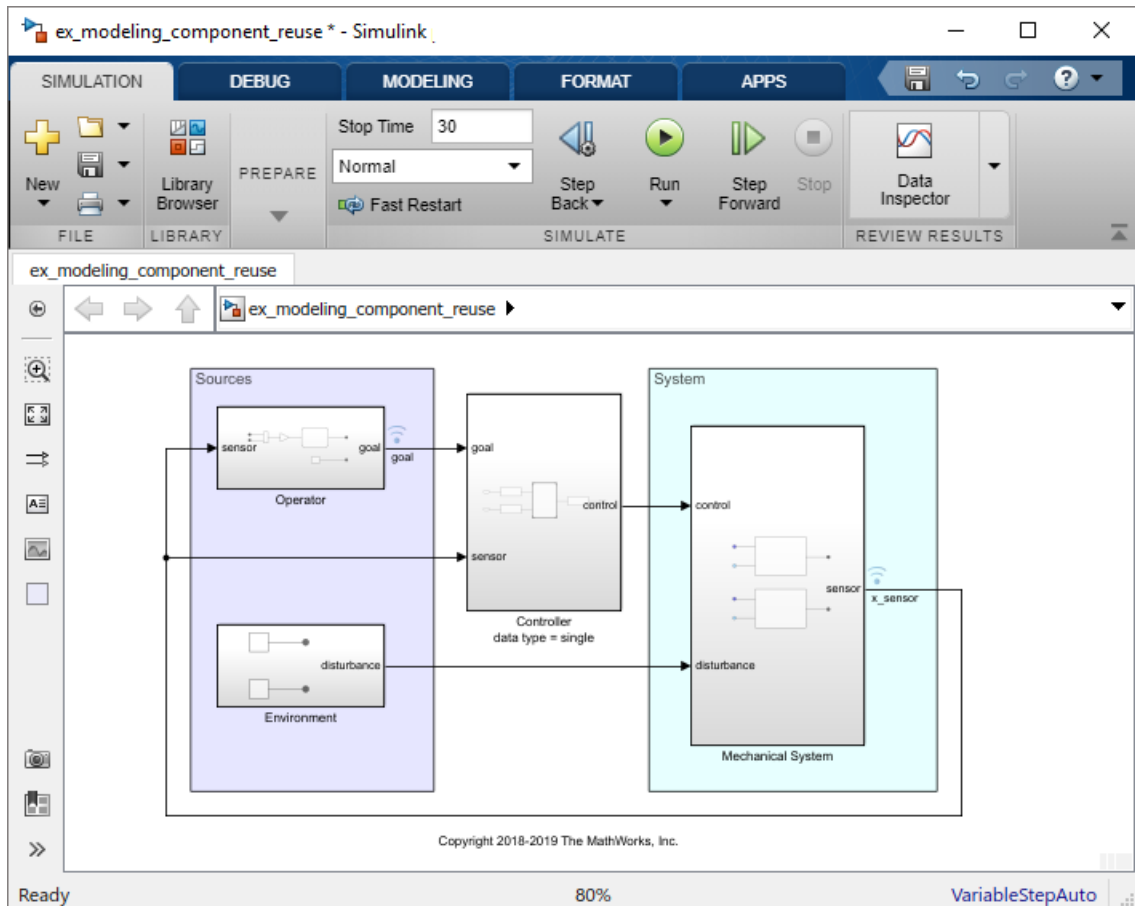


Any subcomponents and ports that are present in the components are deleted when the component links to a Simulink model.

### Export Simulink Model as Architecture

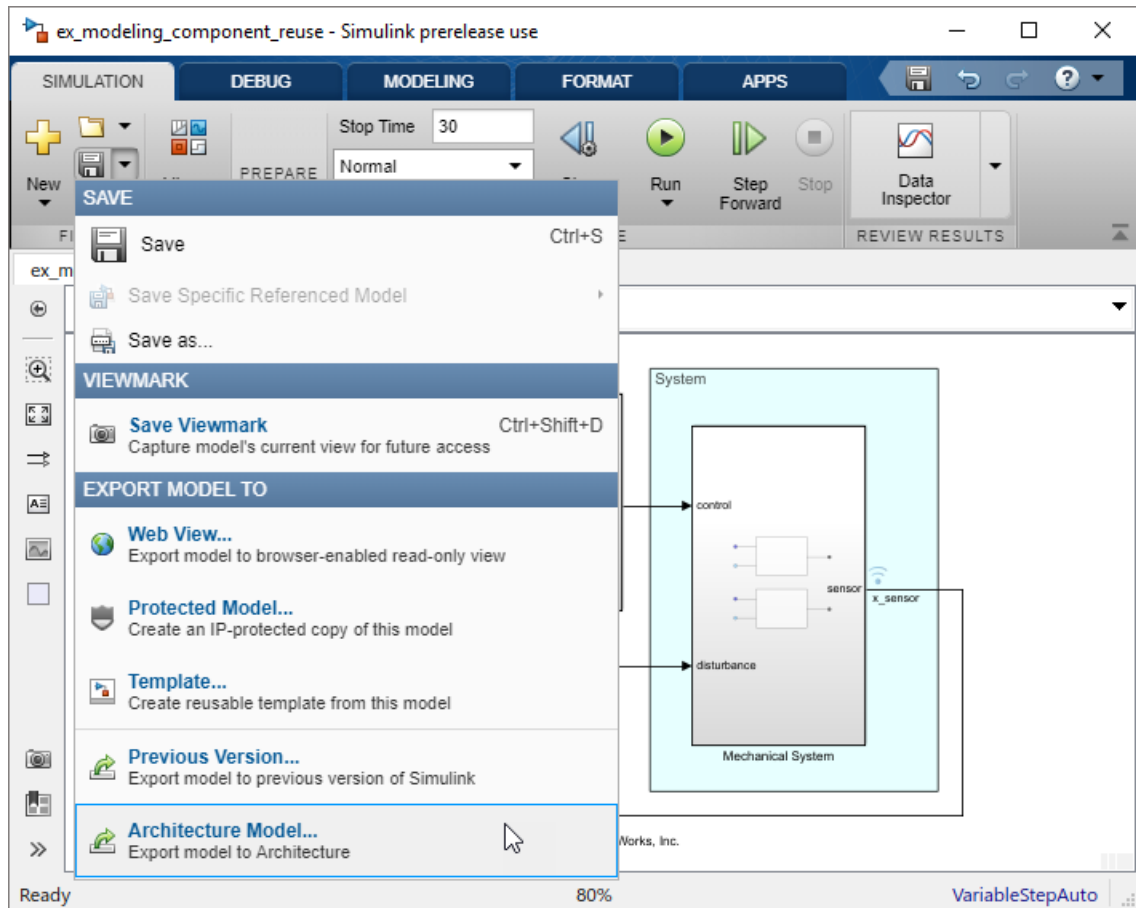
After you extract the architecture from a Simulink model, you can use System Composer architecture editing and analysis capabilities on Simulink models. Model and Subsystem blocks, as well as all ports in a Simulink model represent architectural constructs, while all other blocks represent some kind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

- 1 Open an example model with the MATLAB command `open_system('ex_modeling_component_reuse')`.

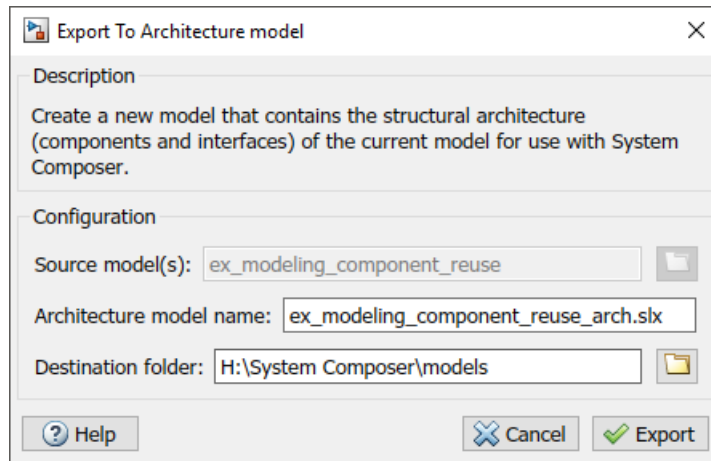


- 2 On the **Simulation** tab, click the **Save** arrow. From the **Export Model To** list, select **Architecture Model**.

## 2 Refactor a Simulink Model as a Composition

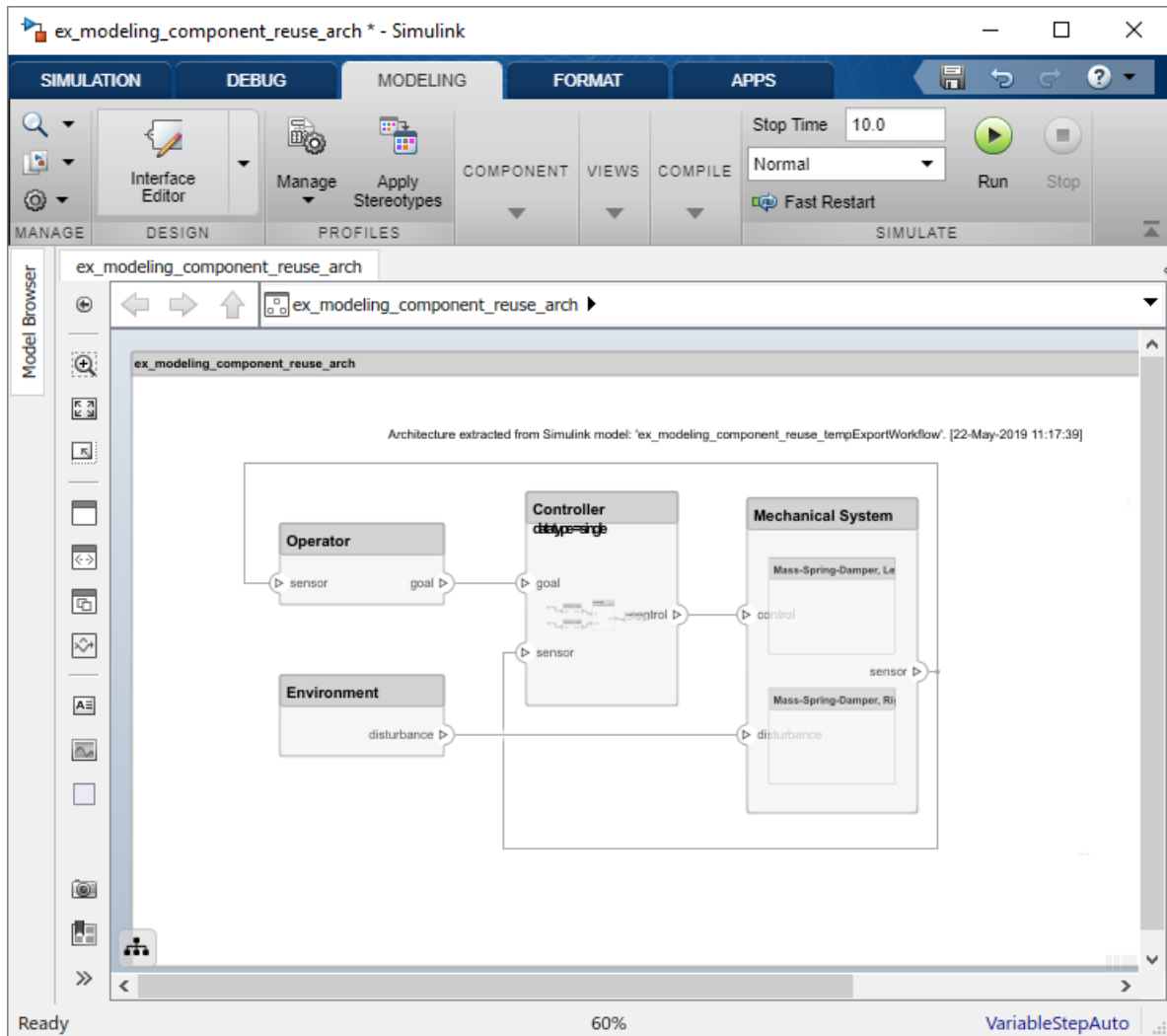


3 Provide a name and path for the architecture model.



- 4 Click **Export**. A System Composer Editor window opens with an architecture model corresponding to the Simulink Model.

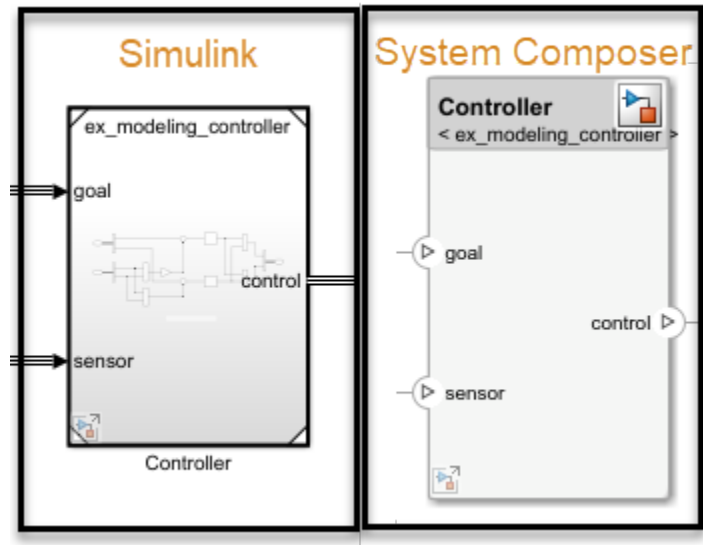
## 2 Refactor a Simulink Model as a Composition



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, nonfunctional properties for model elements and analyze the design.

## See Also

### More About

- "Implement Component Behavior" on page 1-19

