

# Chapter 1 – Quick Start

Ready to start? We are going to dive right in, do a quick pass through the tool, and create a simple SysML model of a doorbell system. Don't worry too much if the SysML concepts presented in this chapter seem strange; we will be back to explain them in more detail in subsequent chapters.

## Introducing the Tool

Before we start constructing our first **model**, there are a few routine housekeeping matters to take care of.

### Starting the Tool

Assuming that you selected the installation option to have *Enterprise Architect* install a Windows desktop icon, the easiest way to start the tool is to click on that icon.

This start page presents a list of recent projects as well as some other options. If you find this start page helpful, feel free to use it. Being somewhat old-fashioned, I generally use the standard “File, Open” interface which is supported as well. As such, I usually click the “X” in the upper right-hand corner of the “Start Page” tab to dismiss the start page.

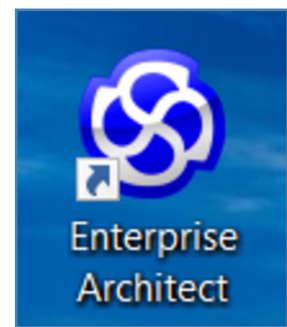


Figure 1-1 – EA icon

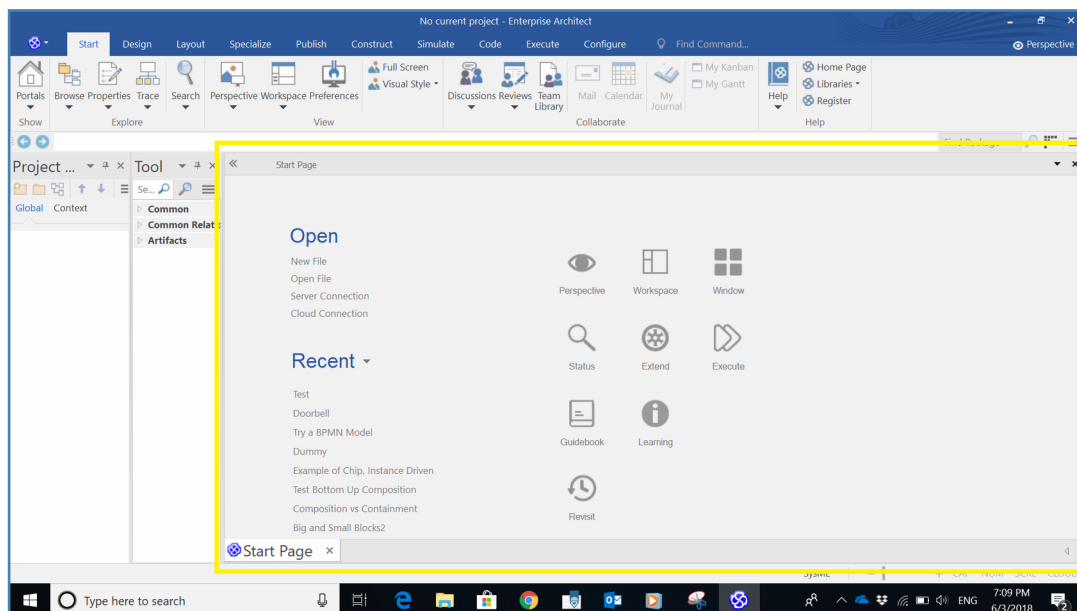


Figure 1-2 – Start Page

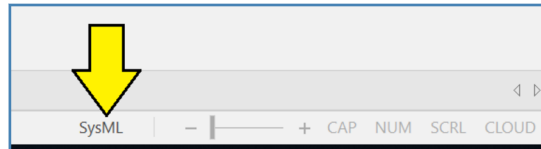


Figure 1-3 – Confirming SysML perspective

Before we get to work creating SysML models, we will want to confirm that *Enterprise Architect* is actually running in SysML mode. Version 14 of the tool has enhanced support for “perspectives”. In previous versions, when you created a new model, the tool would ask you what kind of model you wanted (SysML, BPMN, UML, etc...). In Version 14, the perspective determines what kind of model will be created. You can confirm that the tool is running in the SysML perspective by looking for the “SysML” indicator at the bottom of the screen as shown in Figure 1-3.

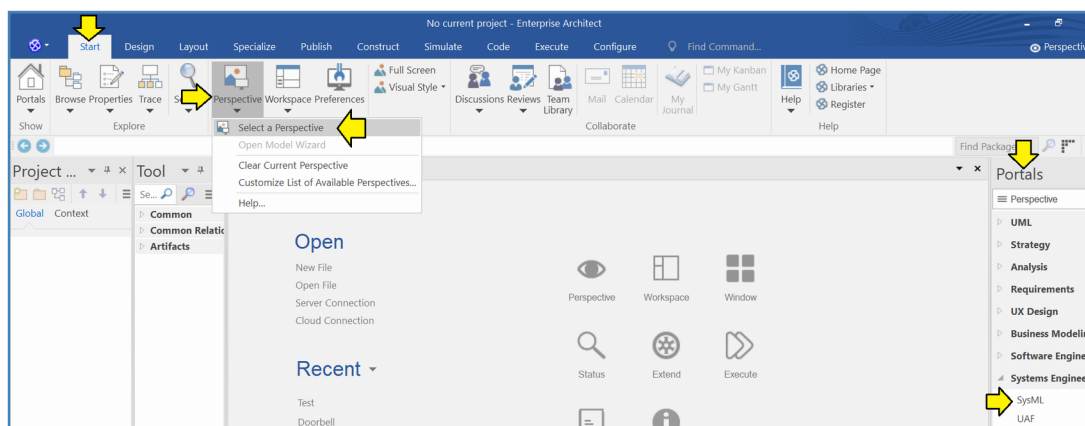


Figure 1-4 – Setting SysML perspective

If the current perspective shows something other than “SysML”, the perspective can be set as shown in Figure 1-4.

- 1) Select the “Start” ribbon.
- 2) Pull down “Perspective” and select “Select a Perspective”.
- 3) The “Portals” pane will appear.
- 4) Under “Systems Engineering” click on “SysML”.

## Backing Up a Model

*Enterprise Architect* version 14 stores each project in a “\*.EAPX” file. This file contains data in a format used by the Microsoft Access JET database engine. It is a binary file, not a text file. The file can be copied, renamed, moved and otherwise manipulated like any other Windows binary file. <sup>(1)</sup>

Likewise, the “\*.EAPX” file can be checked into a version control system like Subversion or GIT as a binary file. Note that since the file is binary, you will not be able to use the differencing functions of the source code control system. However, if you are disciplined about commenting each commit, you will be able to track changes in each version of the project.

## Creating a Model

Now we are ready to create our first model.

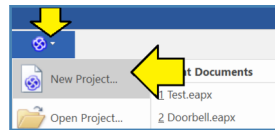


Figure 1-5 – Starting a new project

Pull down the menu in the top left corner of the screen and select “New Project...” to start a new project.

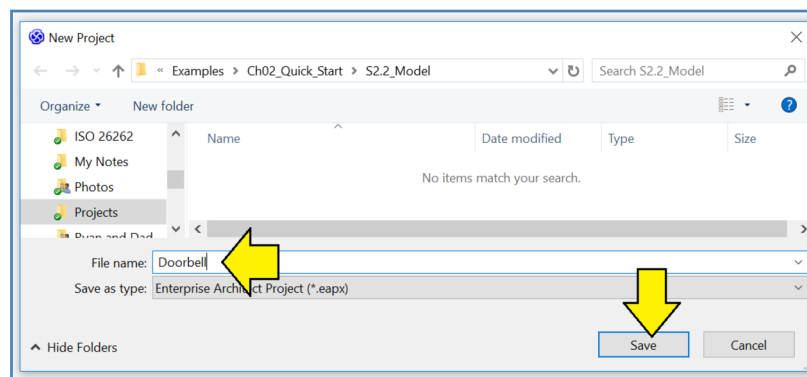


Figure 1-6 – Naming the new project

Give the new project a name. Our first example will be a doorbell system. I recommend that you name the project “Doorbell”.

<sup>(1)</sup> In versions of *Enterprise Architect* previous to version 14, Sparx used the file type “\*.EAP”. In version 14, Sparx introduced the new filetype: “\*.EAPX”.

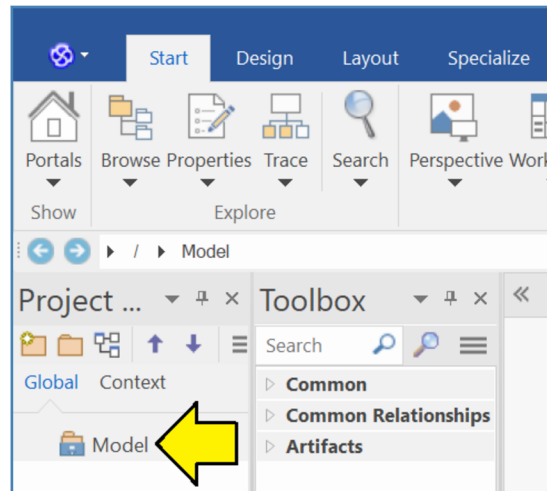


Figure 1-7 – Empty model

We now have an empty SysML 1.5 model. It is possible to use this model as is, but I prefer to give my models descriptive names, especially since we may want to include more than one model in a project later.

In earlier versions of *Enterprise Architect*, you could right-click on the model and rename it. In version 14, there is no such entry in the right-click context menu. However, there is another way to rename the model helpfully pointed out by Sparx support:

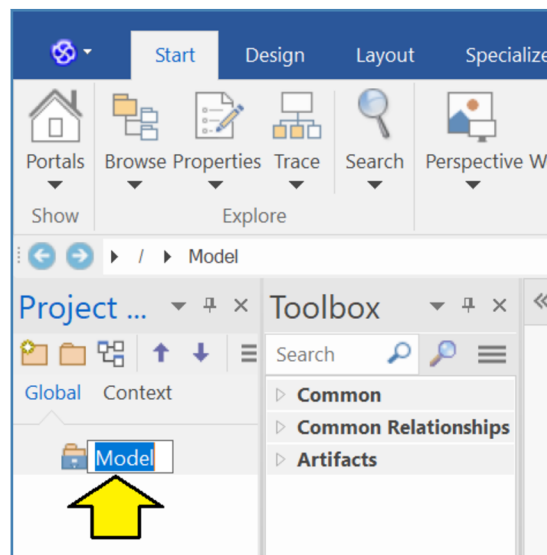


Figure 1-8 – Select model and press F2

Just select the model and press function key 2 (F2). The model will shift into edit mode and you can rename the model. I recommend that you name the model “Doorbell Model” to be consistent with the project.



## Adding a Package

Before we can do anything useful with our new model, we will need to add a **package**. Packages are a central feature of SysML models. In fact, the model itself is a special kind of package. You can think of packages as being very similar to directories in the Windows file system. You can name packages freely. You can put packages within packages.

Let's create our first package.

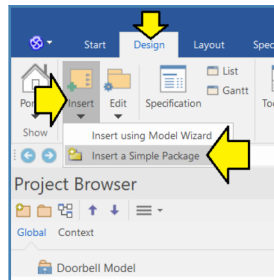


Figure 1-9 – Create the first package

In the current version of *Enterprise Architect*, the first package cannot be created from the right-click context menu. Instead, you have to access the “Design” ribbon, pull down “Insert”, and then select the “Insert a Simple Package” option as shown in Figure 1-9.

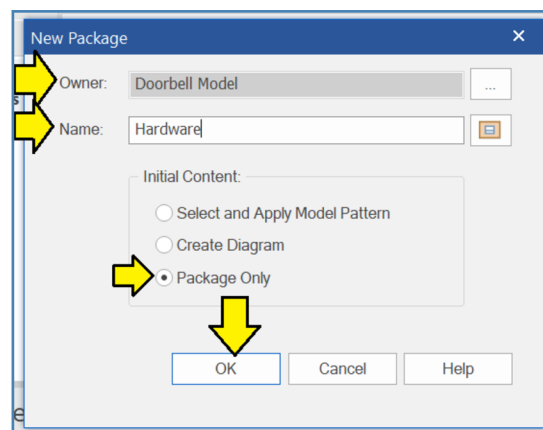


Figure 1-10 – Name the package

- 1) The “Owner:” field should default to the name of the model.
- 2) In the “Name:” field, enter a name for the package. Since we will be modeling the hardware pieces first, I recommend that you name the package “Hardware”.
- 3) For the moment, select the “Package Only” radio button.
- 4) Click “OK” to continue.

## Adding a Block Definition Diagram

Now we are ready to add our first diagram. SysML 1.5 defines nine types of diagrams. The first diagram we will add is a **block definition diagram** – the most basic diagram for defining the elements of a system.

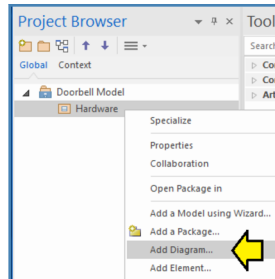


Figure 1-11 – Right-click and add diagram

Right-click on the new package “Hardware” and select “Add Diagram...”.

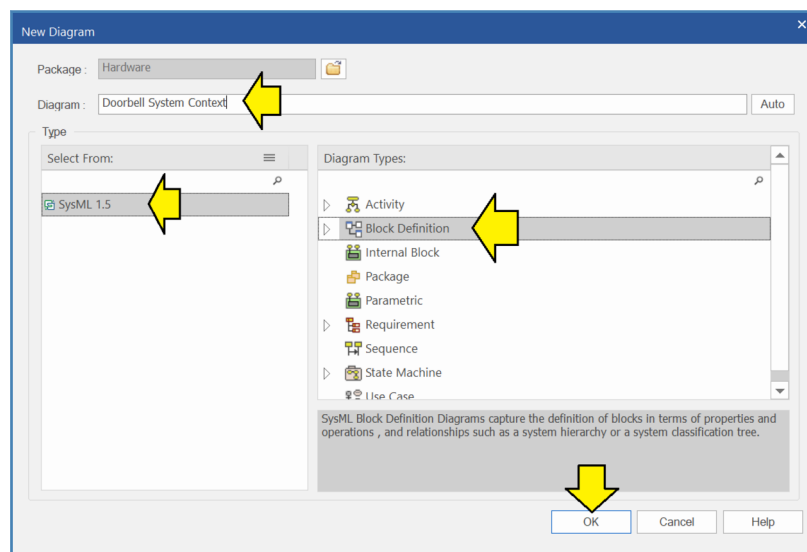


Figure 1-12 – Name the block definition diagram

- 1) Name the diagram “Doorbell System Context”.
- 2) Under “Select From:” select “SysML 1.5”.
- 3) Under “Diagram Types:” select “Block Definition”.
- 4) Click “OK”.

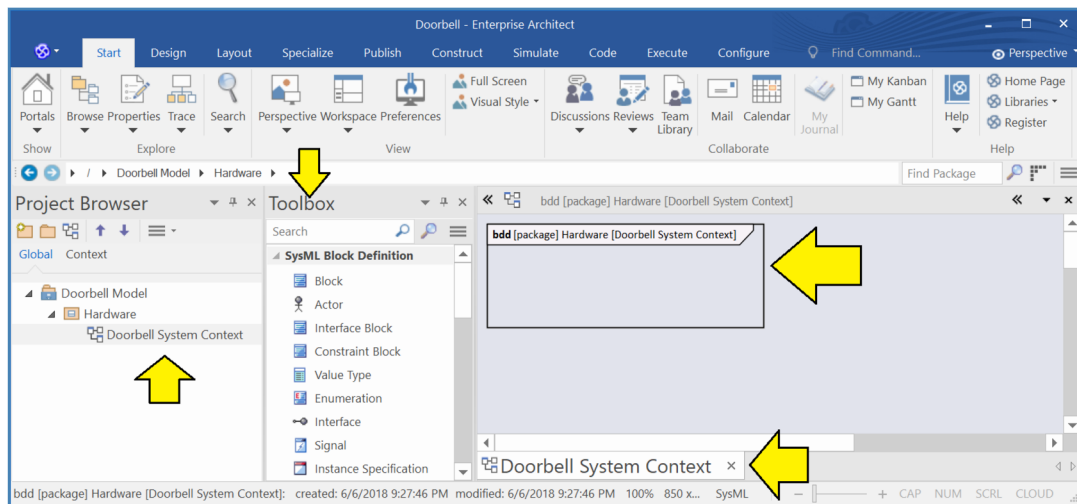


Figure 1-13 – Quick look at the tool

Let's take a quick look at the tool now that we have created the block diagram as shown in Figure 1-13:

- 1) On the left side of the screen, we can see a pane called “Project Browser”. This pane presents a hierarchical view of the model than can be expanded and explored similar to the file explorer user interfaces in Windows and other operating systems.
- 2) In technical articles, this view is often referred to as the “Containment Tree”.
- 3) In the project browser, we see that the tool has added a block diagram called “Doorbell System Context” in the package “Hardware”.
- 4) In the center of the screen, we can see a pane called “Toolbox”. This pane contains icons for different SysML elements that are appropriate for block definition diagrams.
- 5) To the right of the screen, we see that the tool has created our block definition diagram.
- 6) The diagram pane is a “tabbed view” layout which can show multiple diagrams as tabs. We can see the tab for our diagram at the bottom of the screen.
- 7) Our diagram has a standard SysML frame with a header. “bdd” stands for “block definition diagram”. This diagram lives in the package “Hardware”. The name of the diagram is: “Doorbell System Context”.

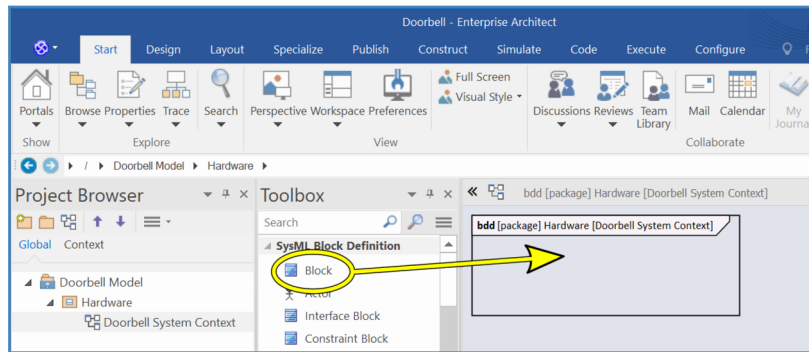


Figure 1-14 – Click on the Block icon and drag it to the diagram

Now we are ready to add our first **block** to our model. In the “Toolbox” pane in the center of the screen, click on the “Block” icon and drag it to the right into the frame of the block definition diagram as shown in Figure 1-14.

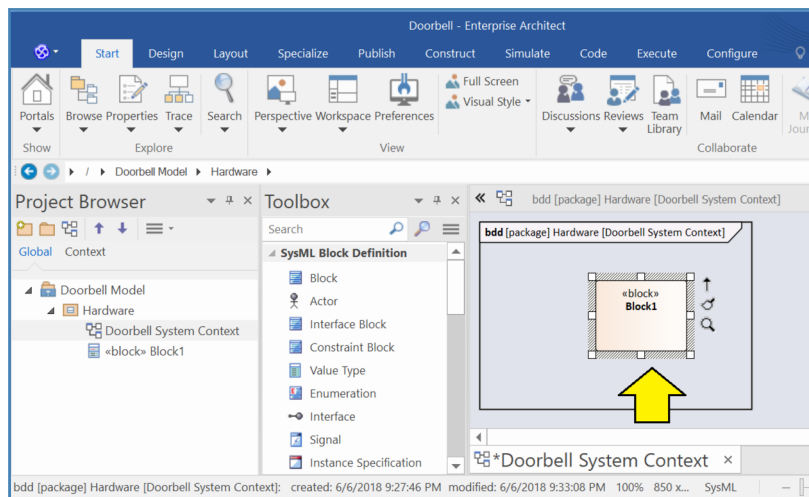


Figure 1-15 – A block will appear in the diagram

A block will appear in the diagram.

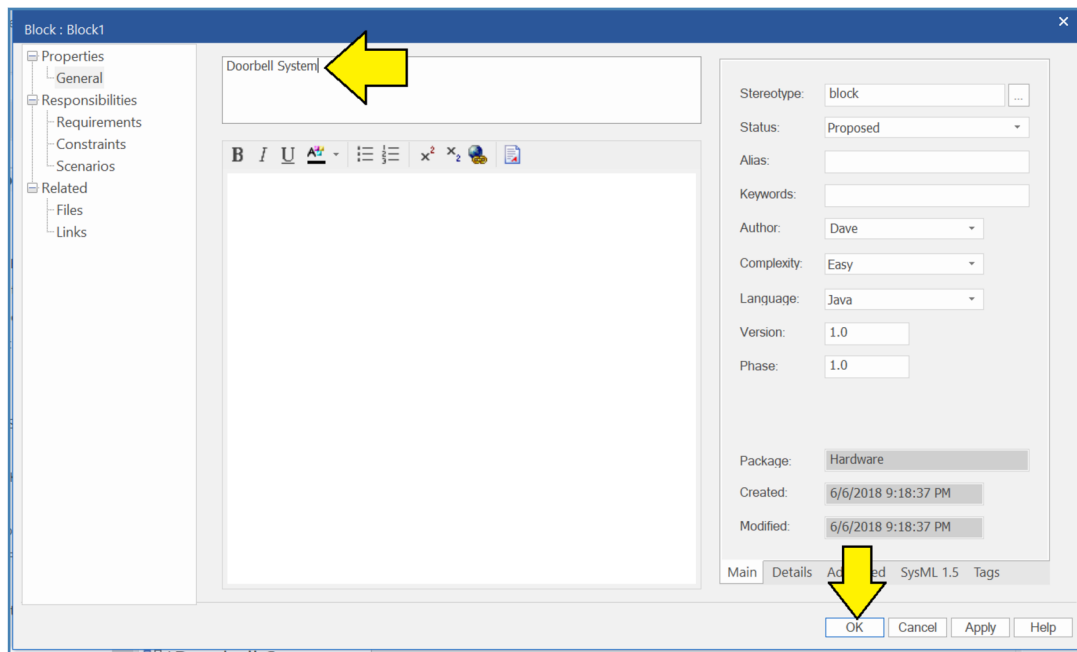


Figure 1-16 – Double-click, name the block, press OK

Double-click on the block. The properties panel for the block will appear. In the top center box, name the block “Doorbell System”. Click “OK”.

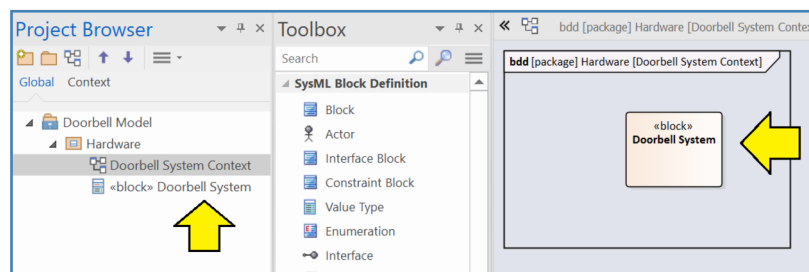
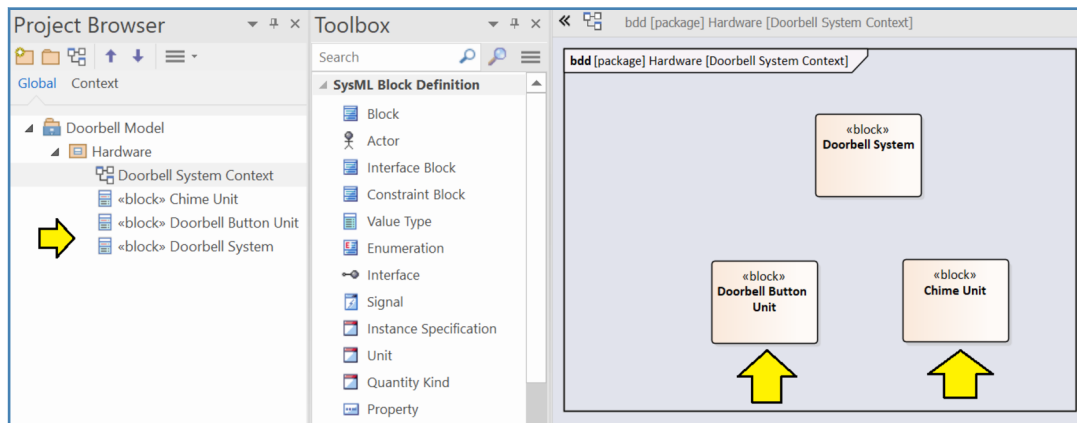


Figure 1-17 – The new block appears in the model at left and in the diagram

Examining Figure 1-17 we see the block in two places. We see the block in the project browser in the left pane. We also see the block in the diagram in the right pane.

### ***Our First Glimpse of Model-Based Systems Engineering (MBSE)***

We are at now at the starting line for “Model-Based Systems Engineering (MBSE)”. That is, the project browser is showing “the model” and the diagram is just a diagram. We could make two or three more diagrams with the same “Doorbell System” block. In the model there would still be one and only one block. That is, the model represents “the truth” and the diagrams are just views of the model. A model element can be shown in as many or as few diagrams as desired. In fact, you can create a model element directly in the project browser and not include it in any diagrams at all. This flexibility to make many different diagrams including the same element becomes a central feature of MBSE because this technique allows the systems engineer to make many simple diagrams for different stakeholders, each diagram showing only the model elements of interest to that stakeholder.



*Figure 1-18 – Add two more blocks to the diagram*

Using the same procedure, add two more blocks and name them:

- Doorbell Button Unit
- Chime Unit

Now that we have the two main units for our system, we are going to introduce a relationship called a **composite association**. This relationship is often referred to more simply as “composition”.

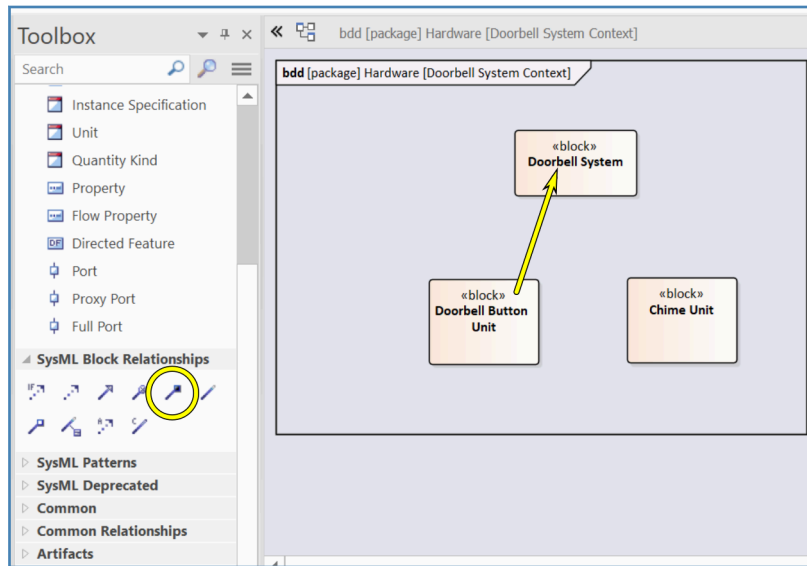


Figure 1-19 – Select the composition relationship and drag as shown

In the toolbox center pane, you will find a section called: “SysML Block Relationships”. Inside this section, you will find an icon that is a line segment connected to a black diamond. Select this icon. Then, drag from the “Doorbell Button Unit” block to the “Doorbell System” block as shown in Figure 1-19.

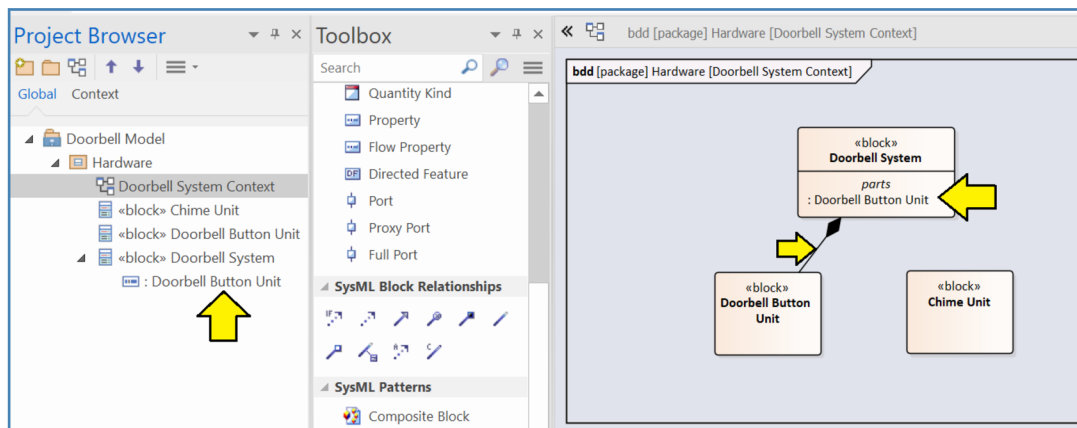


Figure 1-20 – Composition relationship

After you have completed the drag operation to connect the two blocks, three changes will be visible in the tool as shown in Figure 1-20.

- 1) The block “Doorbell System” now shows a compartment called “*parts*”. Inside this compartment is something called: “:Doorbell Button Unit”.
- 2) There is now an arrow whose arrowhead is a black diamond from the “Doorbell Button Unit” block to the “Doorbell System” block.

- 3) In the project browser, we can see that “Doorbell System” now has something nested underneath it called “:Doorbell Button Unit”.

What we have done is modeled the idea that the “Doorbell System” contains a **part** which is a (not yet named) instance of the block of type “Doorbell Button Unit”. For the moment, we are not going to worry too much about the differences between “types” and “instances”. We will cover that topic in more detail in later chapters.

Before we continue, let's give the part a more sensible name. In the project browser, select the part. Press the F2 function key. Name the part “front”. Now we have a part with the qualified name “front:Doorbell Button Unit” which makes a bit more sense. <sup>(2)</sup>

Draw a second composition arrow from the “Doorbell System” block to the “Chime Unit” block and name the new part “kitchen”.

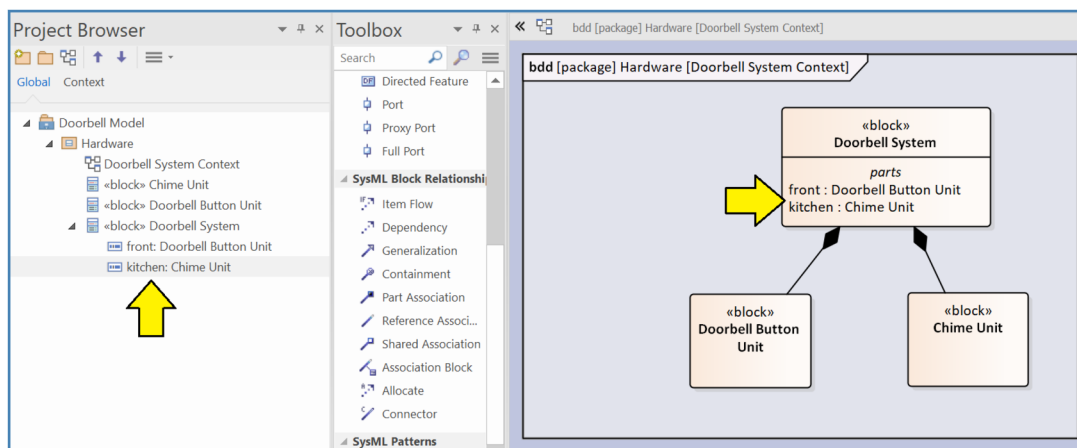


Figure 1-21 – Both part relationships

Your diagram should now look like Figure 1-21.

Next, we will add an **actor** called “Visitor” to the diagram. In the toolbox in the center of the screen you will find an icon for “Actor”.

<sup>(2)</sup> Note that part names are actually “roles” from a UML point of view. We will discuss this in more detail in the chapter about internal block definition diagrams.



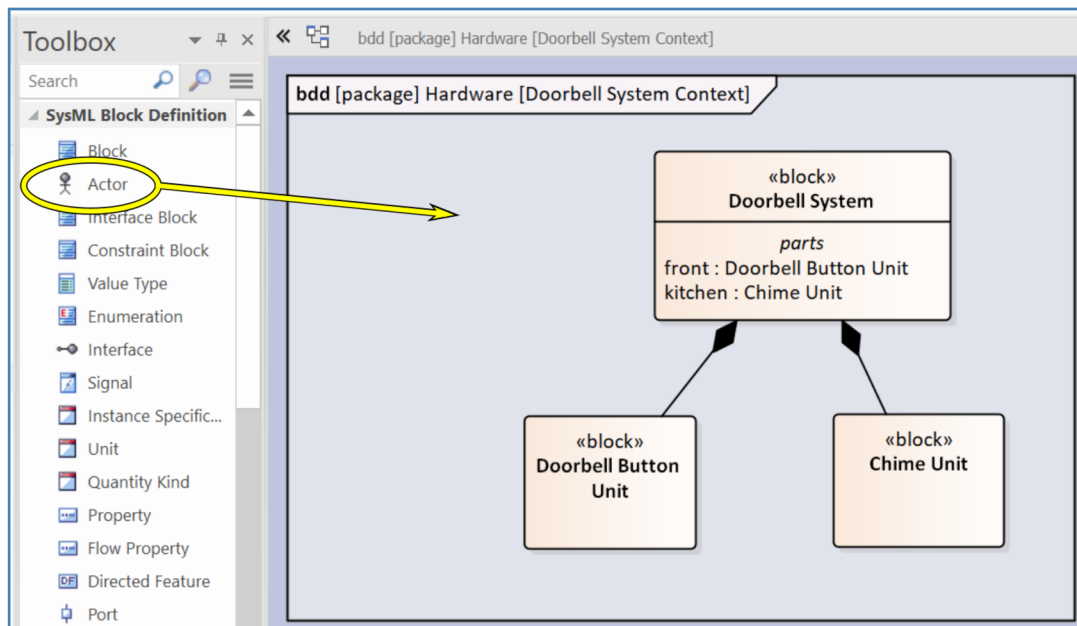


Figure 1-22 – Drag the actor icon to the diagram

Drag the “Actor” icon to the diagram.

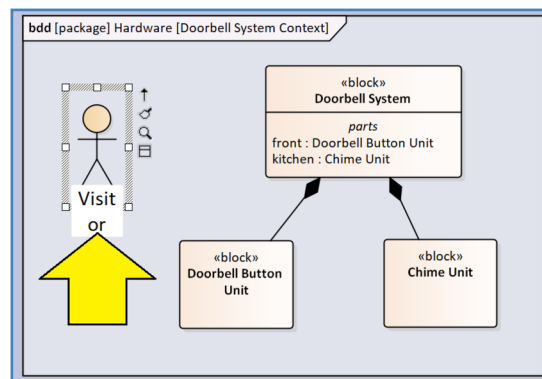


Figure 1-23 – Name the actor

Name the actor “Visitor”.

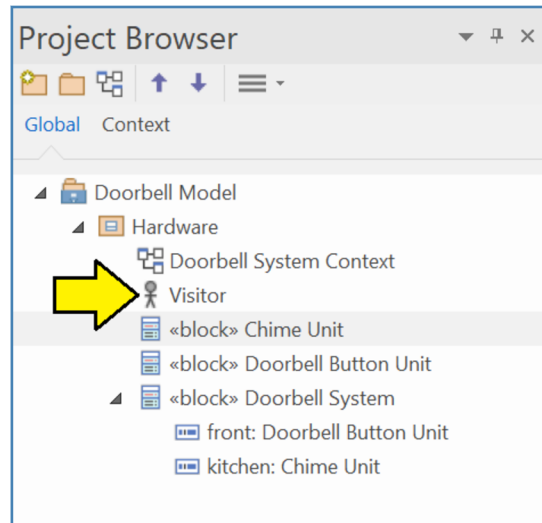


Figure 1-24 – Visitor is now visible in the project browser

We can now see that the visitor is visible in the project browser. However, since our actor is a human<sup>(3)</sup>, we will not want to model the actor as belonging to the package “Hardware”. Add a new package to the model called “Actors”. (See the procedure previously outlined in *Adding a Package* on page 5.)

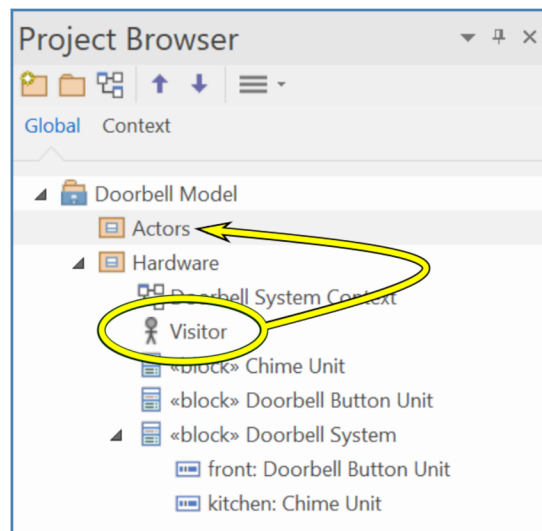


Figure 1-25 – Drag the Visitor to the Actors package

Drag the visitor from the “Hardware” package to the “Actors” package.

<sup>(3)</sup> Actors do not need to be human. In fact, SysML provides an alternate box notation for actors such as cloud server processes that are not humans.

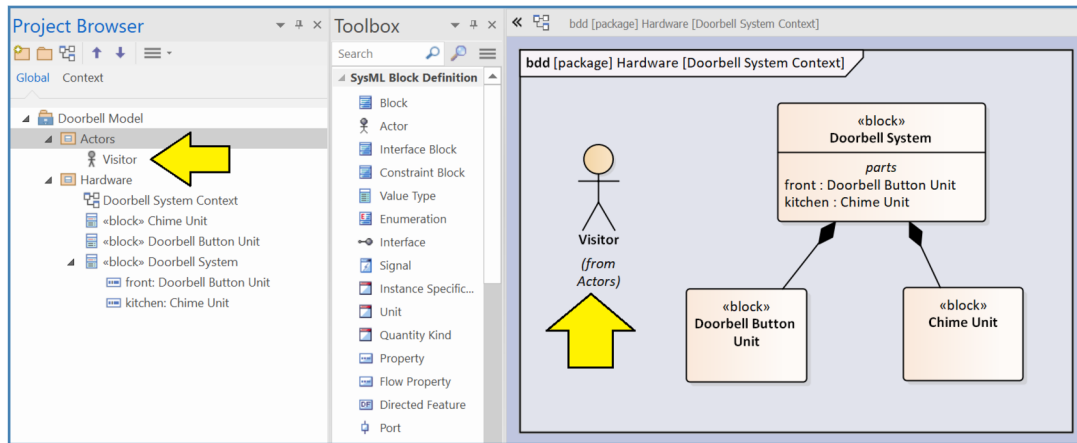


Figure 1-26 – Visitor now in proper package in project browser

Your diagram should now look like Figure 1-26. There are two important points about the diagram in its current form:

- Notice the notation “(from Actors)”. This notation indicates that this element belongs to a different package. That is, the diagram belongs to the package “Hardware” and the visitor is a guest from the package “Actors” so to speak.
- Notice also that the visitor is part of the context for the system but is not part of the system itself. That is, the visitor is shown in the context diagram, but there is no composition arrow from the doorbell system block to the visitor. This point is important: deciding which elements are part of the system and which others are merely things in the neighborhood is one of the first steps in a disciplined systems engineering process.

## Adding a Requirements Diagram

Now we are ready to add a few requirements to our system. SysML provides a **requirements diagram** for this purpose. The first thing we will want to do is add a package to the model called: “Requirements” (See the procedure previously outlined in *Adding a Package* on page 5.)

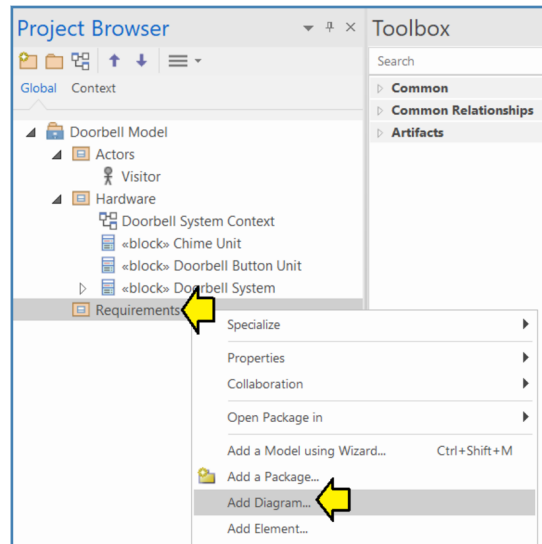


Figure 1-27 – Right-click and select Add Diagram...

Right-click on the “Requirements” package and select “Add Diagram...”.

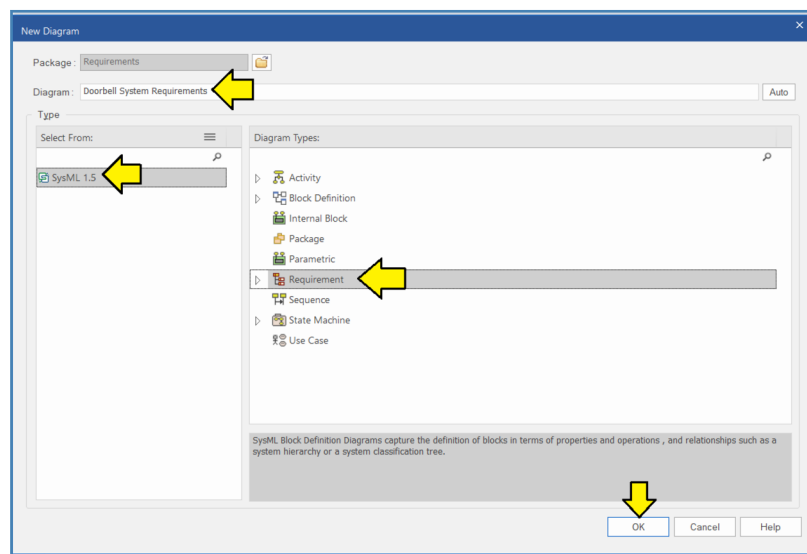


Figure 1-28 – Name the diagram

The “New Diagram” dialog box will appear:

- 1) Name the new diagram “Doorbell System Requirements”.
- 2) Under “Select From:” make sure that “SysML 1.5” is highlighted.
- 3) Under “Diagram Types:” select “Requirement”.
- 4) Click “OK”.

Now that we have created the requirements diagram, we can add a **requirement** to it. This operation is similar to adding a block to a block definition diagram.

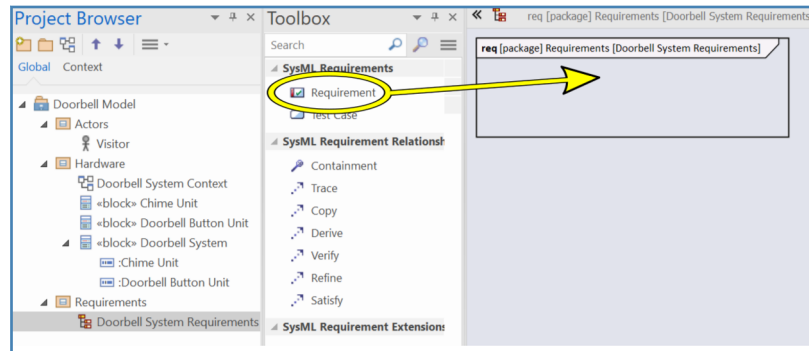


Figure 1-29 – Drag the requirement icon from the Toolbox to the diagram

Drag the requirement icon from the Toolbox to the requirements diagram.

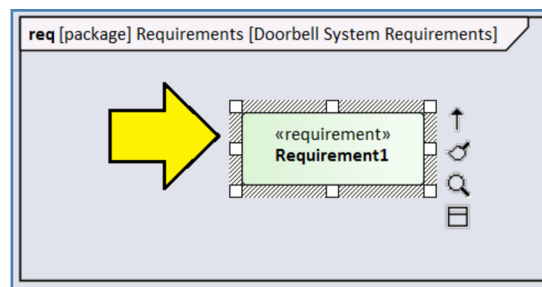


Figure 1-30 – Double-click to open the requirement

We now have a requirement element. Double-click on the element in the diagram to open the properties window for the requirement.

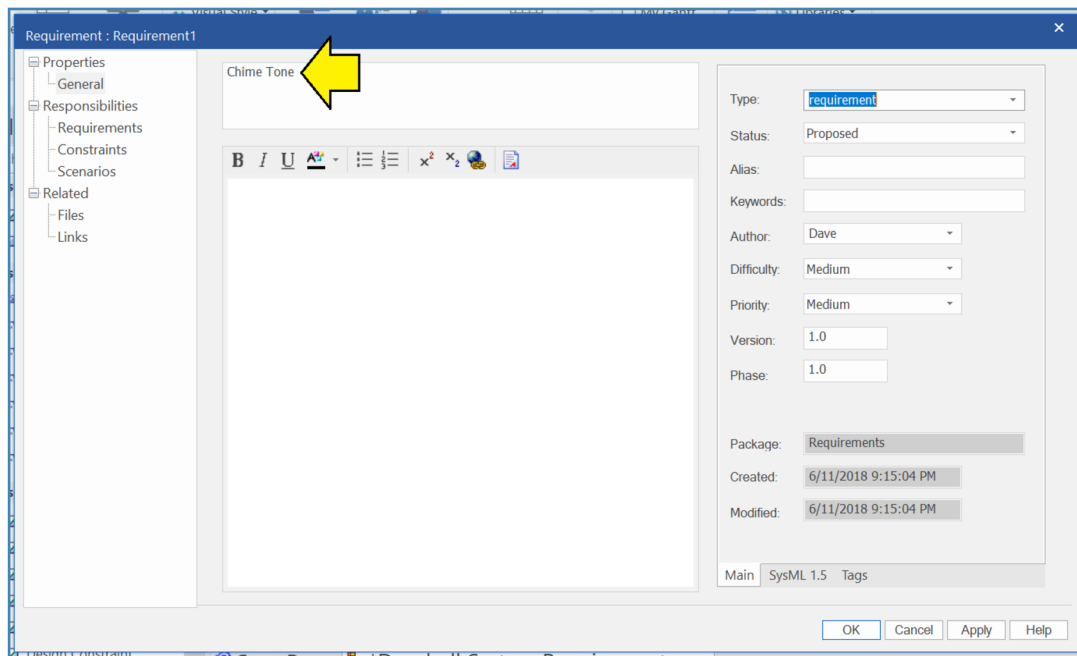


Figure 1-31 – Give the requirement a name

The first order of business is to give the requirement a **name**. Let's call this requirement: “Chime Tone”. Enter the name as shown in Figure 1-31.

The next thing we notice is that it is not immediately obvious where the requirement **id** and **text** should go. <sup>(4)</sup> *Enterprise Architect* is actually older than SysML and was originally designed for UML. Since UML did not explicitly provide for requirements, the makers of *Enterprise Architect* created their own UML extension for requirements and the current SysML requirement feature is built on top of that. As such, the SysML functions appear as sort of an add-on to the original UML extension.

<sup>(4)</sup> An early reader has reported that in version 15 of the tool, the requirements id and text can now be entered directly in the properties panel for the requirement. The steps described here are for version 14 of the tool.

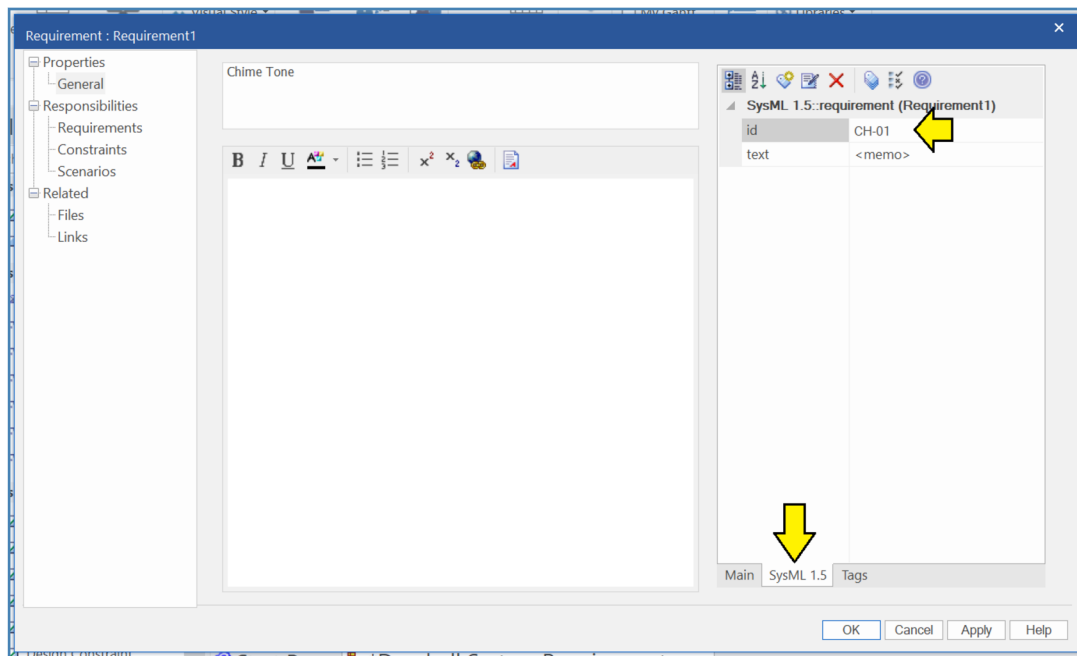


Figure 1-32 – Click the SysML 1.5 tab and assign an ID

In order to set the id and text, first we need to locate and select the “SysML 1.5” tab as shown in Figure 1-32. Once the tab appears, you can enter an id as shown. The id can be any text string. For the moment, let's call this one “CH-01” for “First requirement related to the chime unit”.

We have our requirement id assigned, where does the text go?

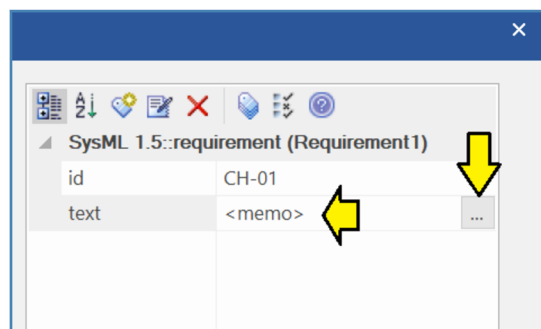


Figure 1-33 – Right-click inside the <memo> field, then click the three dots

- 1) If you look carefully, just under the requirement id field is another field called “text” that contains the text: “<memo>”.
- 2) Click once to the right of “<memo>”.
- 3) The background of the tag “text” will turn dark and a new small box with three dots in it will appear to the right of “<memo>”.
- 4) Click on the three dots.

- 5) A dialog box will appear.

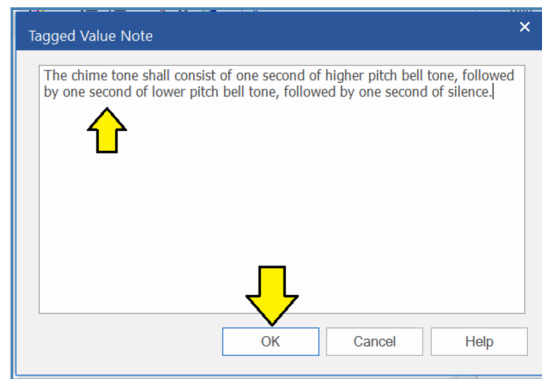


Figure 1-34 – Enter the text of the requirement

Enter the following text:

*The chime tone shall consist of one second of higher pitch bell tone, followed by one second of lower pitch bell tone, followed by one second of silence.*

When you are done, click “OK”.

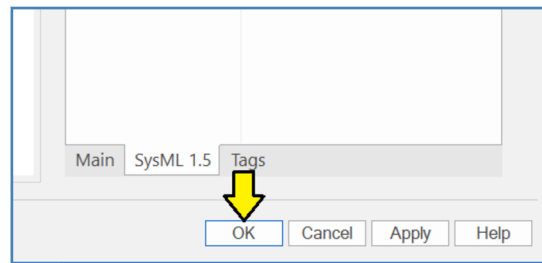


Figure 1-35 – Click OK to finish

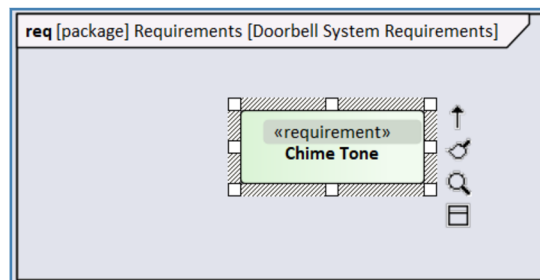


Figure 1-36 – By default the diagram does not show id or text

After you click “OK” your diagram will look like Figure 1-36. Where are the id and text? In *Enterprise Architect* these are not shown by default.



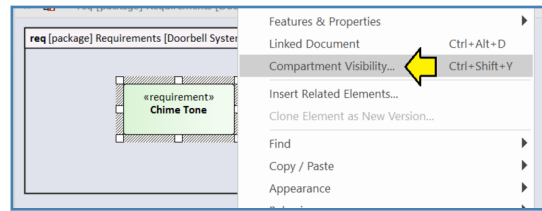


Figure 1-37 – Right-click and select compartment visibility

Right-click on the requirement in the diagram and select “Compartment Visibility...”.

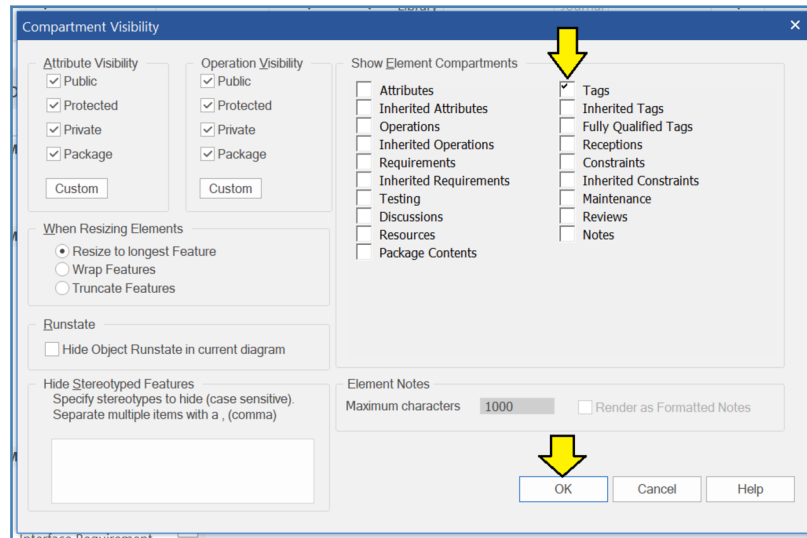


Figure 1-38 – Select Tags and click OK

*Enterprise Architect* considers the id and text to be “Tags” that have been added to its requirement element. In order to make the id and text show up, select “Tags” and click “OK” as shown in Figure 1-38.

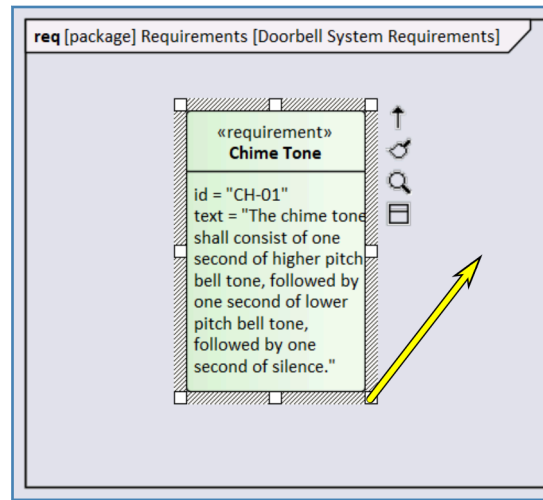


Figure 1-39 – Drag the corner of the requirement box to resize it

After you have set the requirement element in the diagram to display the id and text you will probably notice that the overall shape of the requirement box is probably not ideal. Also, the text may be bumping up against the border of the box. You can simply grab a corner of the box in the diagram with the mouse and drag to resize it. The tool will reflow the text as you resize the box.

#### ***Tip - Resizing the Diagram***

What if you want to zoom the diagram in or out? You can resize any diagram and zoom in or out by holding down the control key and rolling the wheel on your mouse. Actually, there are three motions (which are common to some other graphical software packages):

- control key + mouse wheel = zoom
- shift key + mouse wheel = move left or right
- (nothing) + mouse wheel = move up or down

Let's add a few more requirements. Have you ever visited someone's house and been frustrated because it wasn't clear whether the doorbell button was really doing anything or not? Our doorbell system is going to solve that problem:

- 1) As the visitor approaches the door, the doorbell button will be glowing dimly.
- 2) After the visitor pushes the button, the chime will sound inside, and the doorbell button will begin flashing brightly on and off.
- 3) When the chime is finished, the doorbell button will return to the glowing dimly state. <sup>(5)</sup>

<sup>(5)</sup> The observant reader will notice that we have just created the beginnings of a “user story” for our system. User stories are an important part of almost any design methodology.

In order to save you time and effort in typing, I have provided a spreadsheet of these requirements in the example files in the directory *Examples\Cb02\_Quick\_Start\S2.5\_Req\_Diag*

| ID    | Name                           | Description   |
|-------|--------------------------------|---|
| CH-01 | Chime Tone                     | The chime tone shall consist of one second of higher pitch bell tone, followed by one second of lower pitch bell tone, followed by one second of silence. |
| CH-02 | Chime Cycle                    | The chime cycle shall consist of three chime tones.   |
| CH-03 | Chime Cycle After Button Press | When the doorbell button is pressed, the chime unit shall complete one chime cycle.   |
| BT-01 | Flash During Chime             | While the chime cycle is in progress, the doorbell button shall flash.  |
| BT-02 | Flash Cycle                    | The button flash cycle shall be 0.5 second bright illumination followed by 0.5 second of no illumination.   |
| BT-03 | Idle Dim Glow                  | When the doorbell system is idle, the doorbell button shall glow dimly.   |

Figure 1-40 – Requirements for the doorbell system

Go ahead and repeat the procedure and add the remaining five requirements to the model.

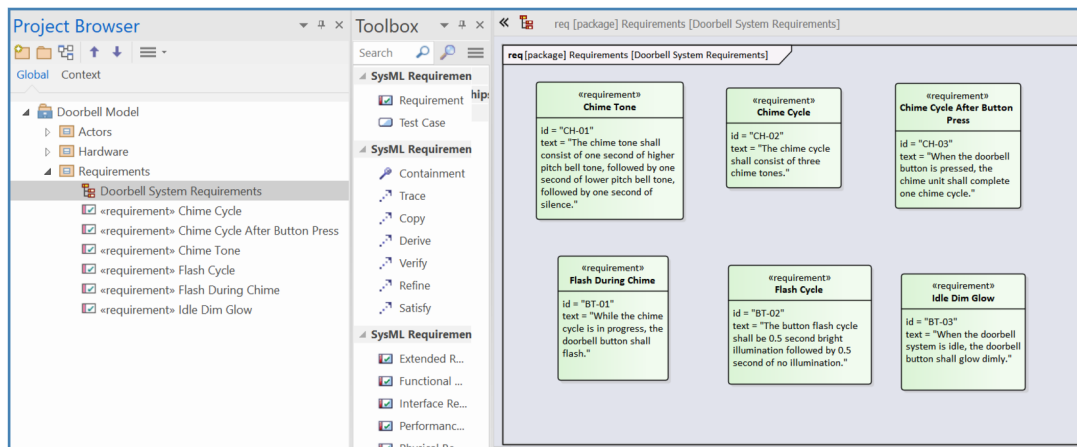


Figure 1-41 – Completed requirements for the doorbell system

When you are done adding the requirements, your requirements diagram should look like Figure 1-41.

So far, we have shown how to get a set of requirements into the system and how to place them on a diagram, but we really haven't yet shown the compelling benefit of integrating requirements into a graphical system model. In order to show this benefit, create a second requirements diagram called “Button Glow Detail”.

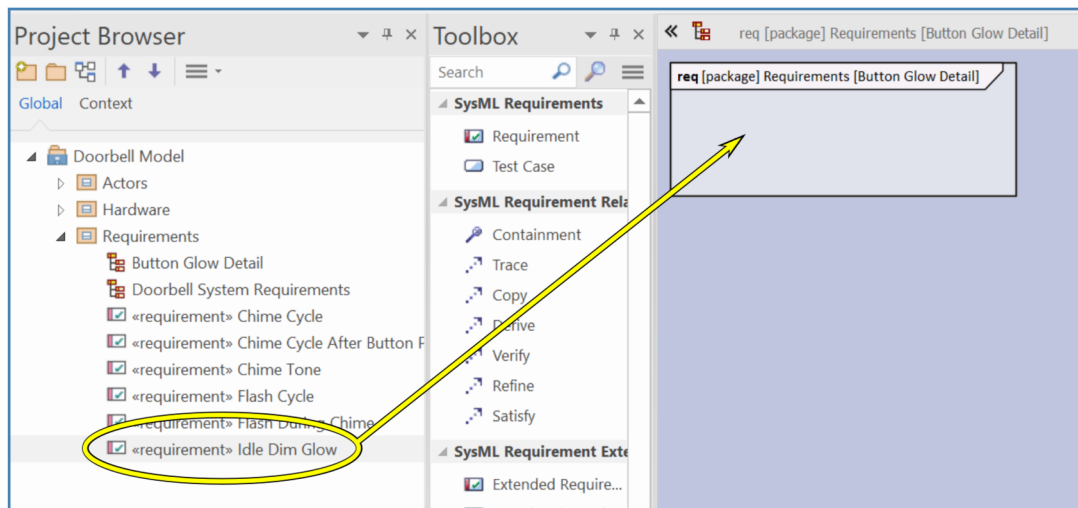


Figure 1-42 – Drag the idle dim glow requirement to the diagram

Drag the “Idle Dim Glow” requirement to the diagram. Select the default of “Drop as Link”.

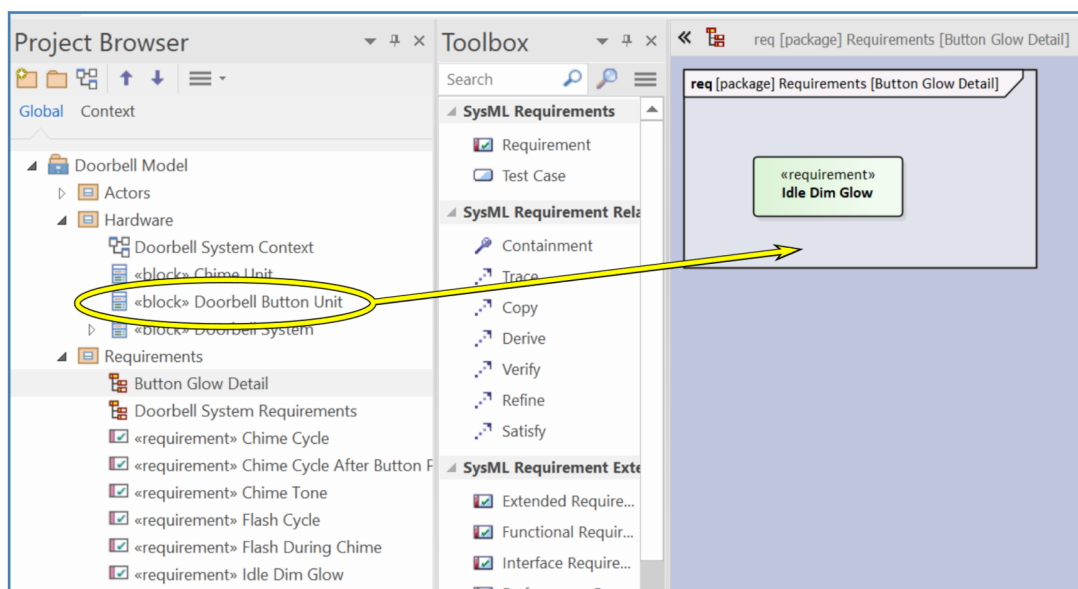


Figure 1-43 – Drag the doorbell button unit block to the diagram

Next drag the “Doorbell Button Unit” block to the diagram. Select the default of “Drop as Link”.

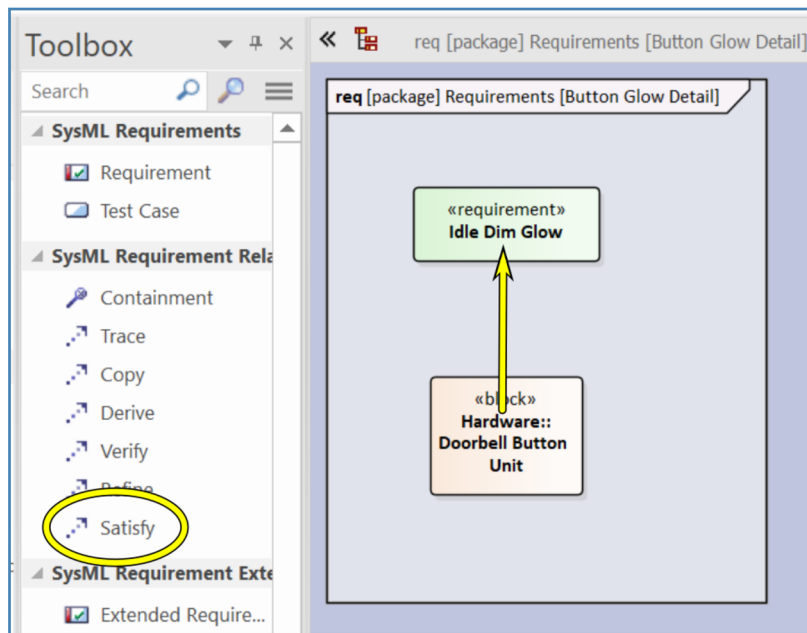


Figure 1-44 – Select satisfy and drag relationship

In the toolbox select the “Satisfy” icon and drag a **satisfy** relationship between the button unit and the requirement.

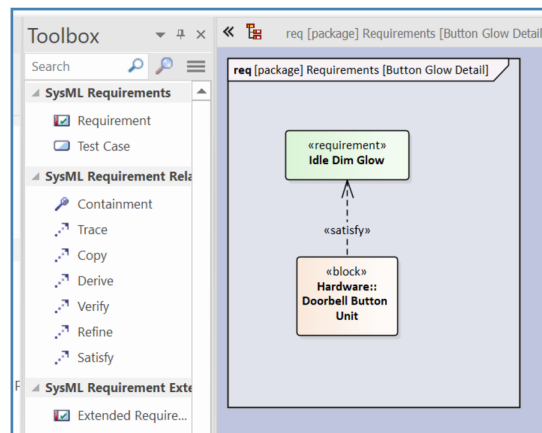


Figure 1-45 – Completed satisfy relationship

When you are done adding the satisfy relationship, your requirements diagram should look like Figure 1-45. This diagram can be understood as meaning: “The doorbell button unit is responsible for satisfying the idle dim glow requirement”.

Quickly examining the toolbox, the reader will notice that there is also a SysML element for test case as well as several more descriptive relationships. In the chapter on requirements diagrams, we will go into more detail about how these relationships can be used to make helpful diagrams that help stakeholders understand complex relationships between requirements, system elements, and test cases.

### ***Direction of Arrows***

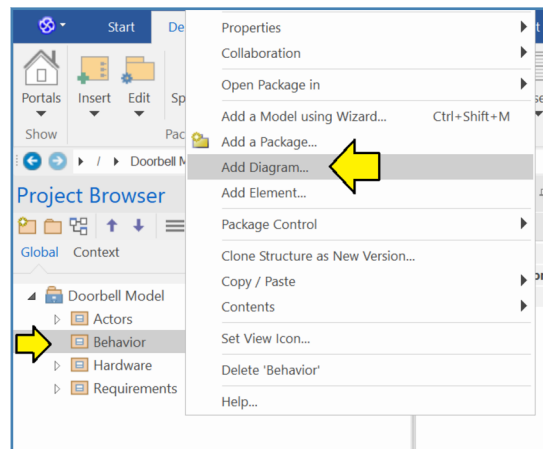
When first shown a diagram like Figure 1-45, many engineers will feel like the arrow is pointed in the wrong direction. Such engineers are used to thinking of requirements as flowing downward from top to bottom. SysML inherits its arrow direction philosophy from UML. One point about UML (and SysML) is that arrows are navigated in the direction that they point. Hence, the requirement relationship arrows are “owned” by the elements that need to satisfy the requirements. This arrangement makes it slightly easier later to answer the question: “Which requirements does this block need to satisfy?”

## **Adding a Sequence Diagram**

In the preceding sections, we have taken a quick look at diagrams for structure and for requirements. The last thing we will do in this chapter is add a diagram that describes system behavior. SysML defines three behavioral diagrams:

- Activity diagram
- Sequence diagram
- State machine diagram

In this section, we will create a simple **sequence diagram** for our doorbell system.



*Figure 1-46 – Add a package and a diagram*

Add a new package for “Behavior” to the model. Right-click on the new package in the project browser and select “Add Diagram...”.

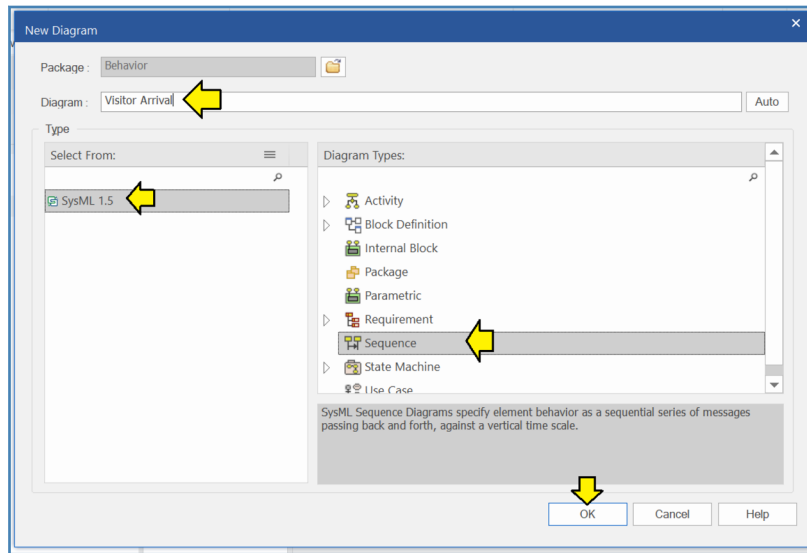


Figure 1-47 – Name and create the diagram

Give the diagram a name. The sequence diagram is actually the internal representation of a model element known as an **interaction**. As such, it makes sense to name a sequence diagram with a noun phrase that represents an interaction or activity. Name this diagram: “Visitor Arrival”. Click “OK” when you are done.

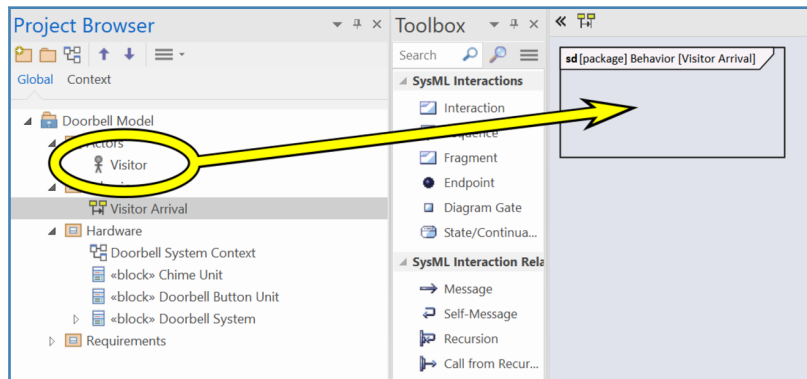


Figure 1-48 – Drag Visitor to the diagram

Drag the visitor from the project browser to the diagram.

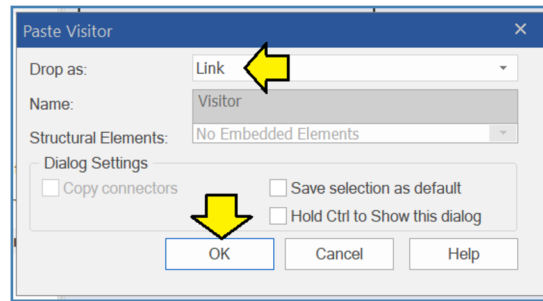


Figure 1-49 – Accept link

A dialog box will appear confirming that you want to create a “Link” to the diagram (as opposed to creating an instance or an object which are not what we want to do right now). Accept the “Drop as:” default value of “Link” and click “OK”.

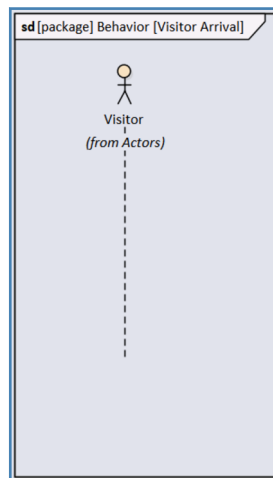


Figure 1-50 – Lifeline

After you click “OK” *Enterprise Architect* will create something called a **lifeline** for the Visitor element.

- At the top is the element itself.
- Underneath this is a notation of the package which owns the element
- Below that is a dashed line. The dashed line indicates the passage of time in the “life” of the element, starting at the top and progressing downward in time.



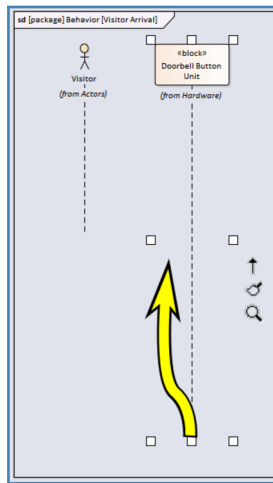


Figure 1-51 – Drag box to shorten lifeline

Next drag the block named “Doorbell Button Unit” from the project browser to the diagram.

Oops! The lifeline is a little longer than we need. You can shorten or lengthen a lifeline by clicking on it and dragging the boxes around as shown in Figure 1-51.

Drag the block named “Chime Unit” to the diagram as well.

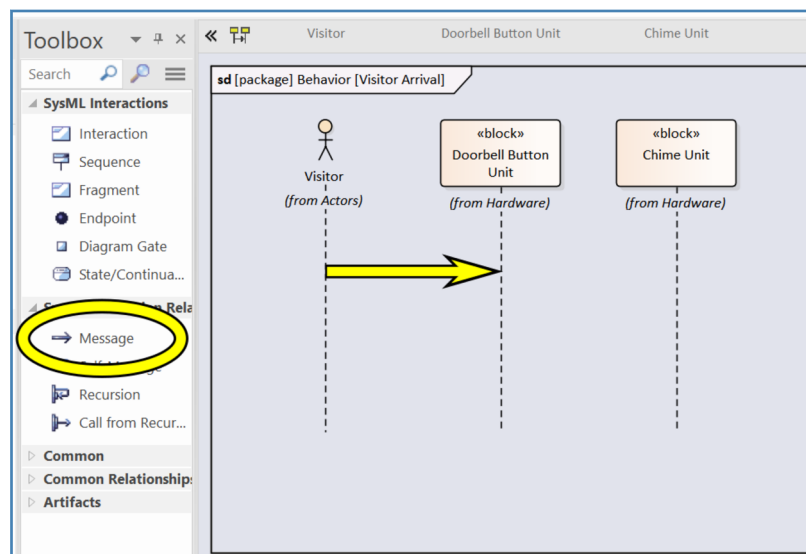


Figure 1-52 – Drag a Message from Visitor to Doorbell Button Unit

We now have three lifelines for three elements. Now we are ready to start creating the actual interaction between these three elements. In a sequence diagram each communication between two lifelines is a **message**. In the toolbox select “Message” and drag from the lifeline for the visitor to the lifeline for the doorbell button unit.

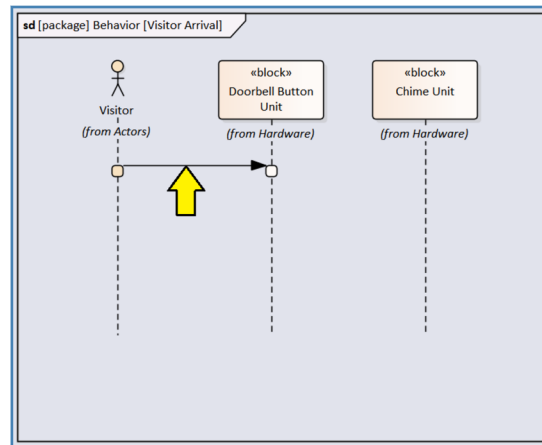


Figure 1-53 – Double-click on the message

We now have our first message on the diagram. However, there are two things we will want to change immediately:

- The message needs a name.
- The solid black line and solid black arrowhead represent the default message type of “synchronous”. Unless we expect the visitor to stand with finger attached to the button until the doorbell chime is complete, we probably don't want to model this message as a synchronous message. An asynchronous message will be more appropriate.

Double-click on the message arrow to open the properties window.

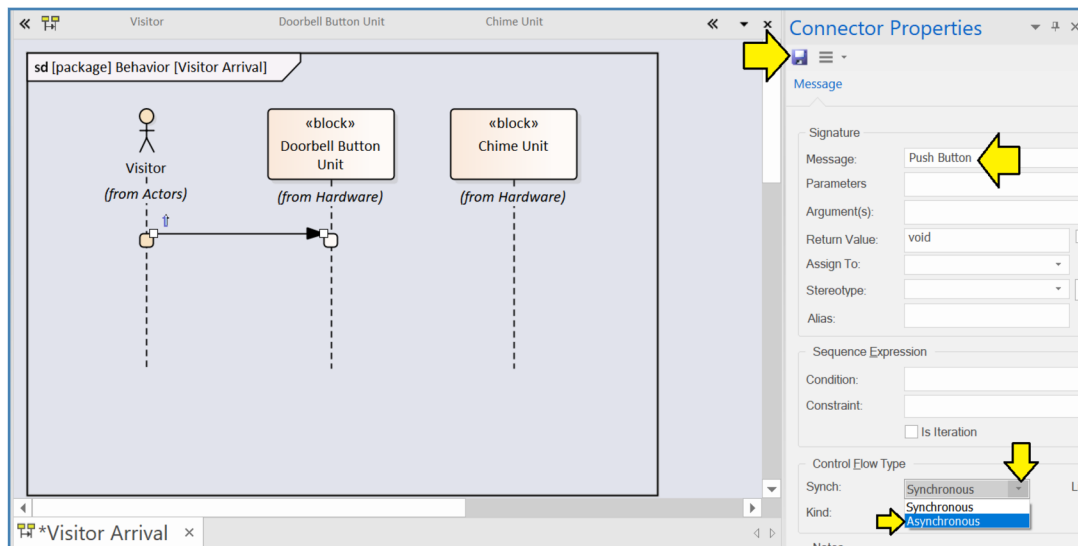


Figure 1-54 – Name message

A window called “Connector Properties” will appear to the right of the sequence diagram. This window is new in *Enterprise Architect* version 14.

- 1) Name the message: “Push Button”
- 2) Pull down “Synch:” under “Control Flow Type” and select: “Asynchronous”.
- 3) In order to save your changes, click on the floppy disk icon in the upper left corner of the “Connector Properties” window.

Now let's go ahead and add some additional asynchronous messages to complete the modeling of our story:

- 1) Add a message called “Button Pushed” from the button to the chime.
- 2) Add a message called “Chime Starting” from the chime to the button. Later, as we add more detail to the model, this will become the trigger to make the button begin flashing brightly.
- 3) Add a message called “Button Flashing” from the button to the visitor. The flashing is just a visual indication, but it is a message nonetheless and we can model it as an asynchronous message.
- 4) Add a message called “Chime Ending” from the chime to the button. This will become the trigger to return the button to “dull glow” mode.
- 5) Add a message called “Button Dull Glow” from the button to the visitor. The sudden absence of flashing is itself a message and can be modeled as such.

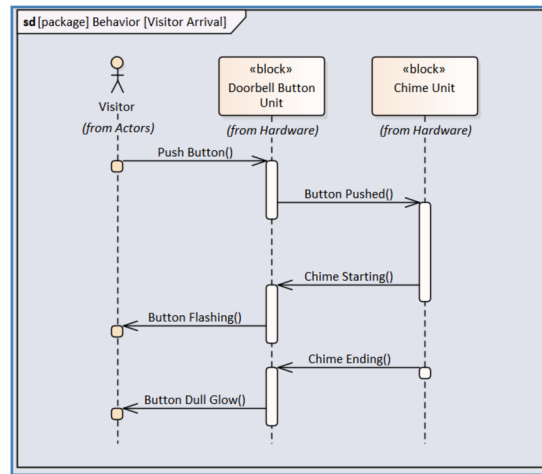


Figure 1-55 – Finished sequence diagram

Your finished sequence diagram should look like Figure 1-55.

This concludes our quick look at SysML and at the *Enterprise Architect* tool. In the next chapter, we will cover some general topics and questions before we start looking at each of the nine SysML diagram types in more detail.