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 come 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 *Cameo Systems Modeler* install a Windows desktop icon, the easiest way to start the tool is to click on that icon.

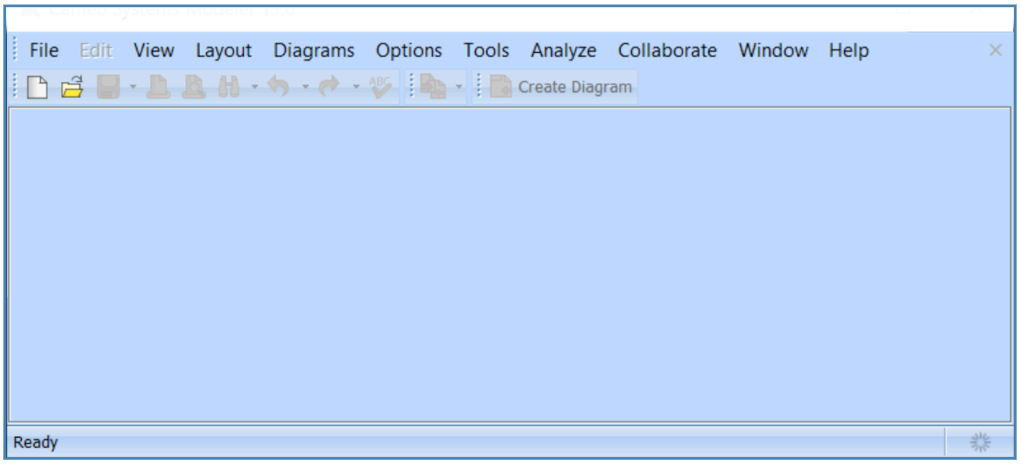


Figure 1-1 – Start page

This start page presents the standard "File, Open" interface.

Backing Up a Model

Cameo Systems Modeler stores each project in a "*.mdzip" file. This file is a binary file, not a text file. The file can be copied, renamed, moved and otherwise manipulated like any other Windows binary file.

The "*.mdzip" 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. NoMagic also offers a cloud server for collaboration called *Teamwork Cloud.* [(1)]

Creating a Model

Now we are ready to create our first **model**.

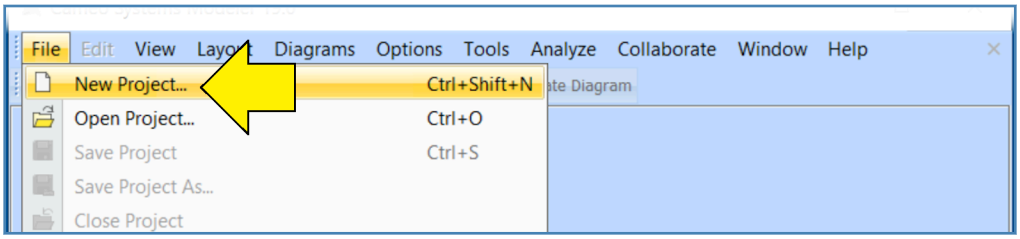


Figure 1-2 – Starting a new project

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

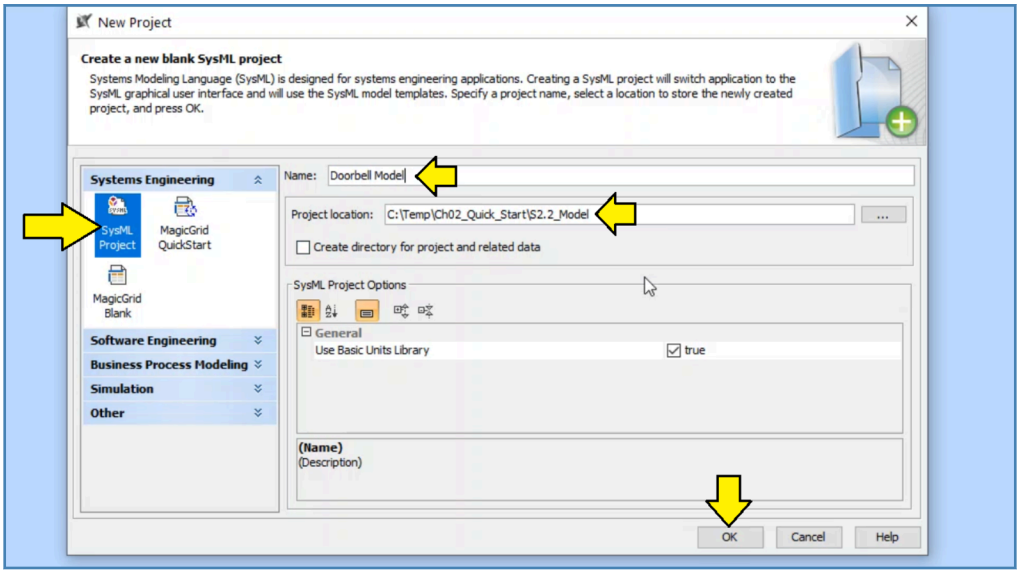


Figure 1-3 – Naming the new project

Make sure that the project type "SysML Project" is selected. Choose a directory for the project. Give the new project a name. Our first example will be a doorbell system. We recommend that you name the project "Doorbell Model".

(1) See: https://www.nomagic.com/products/teamwork-cloud

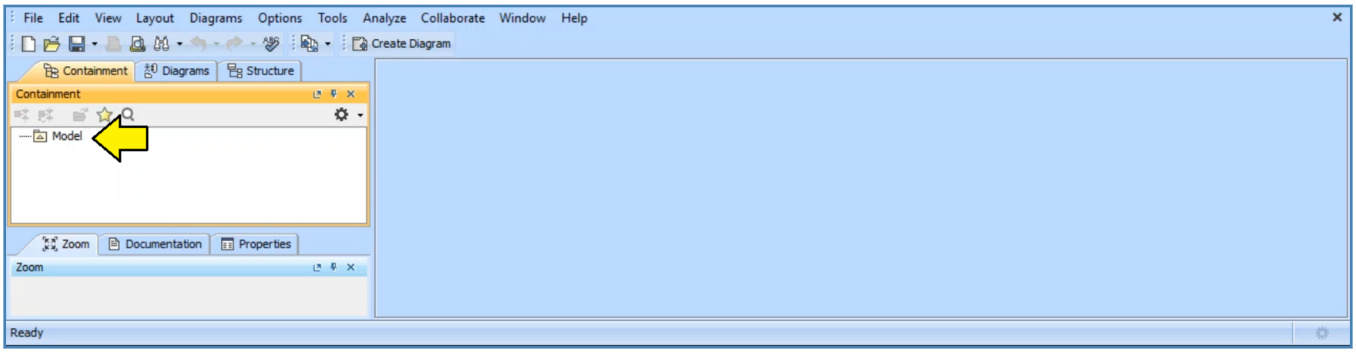


Figure 1-4 – Empty model

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

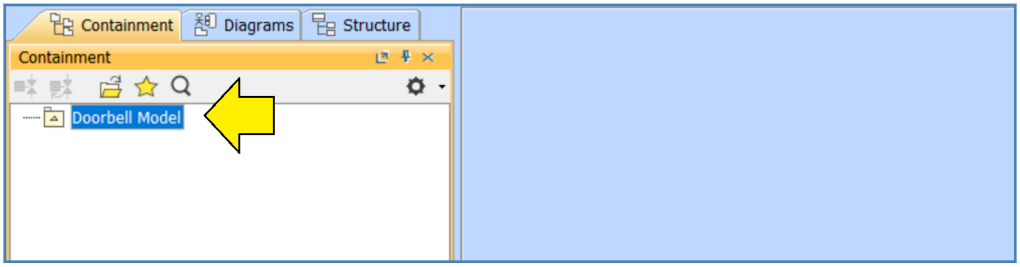


Figure 1-5 – Select model and press F2

Just select the model and press function key 2 (F2). The model will shift into edit mode and you can rename the model. 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.

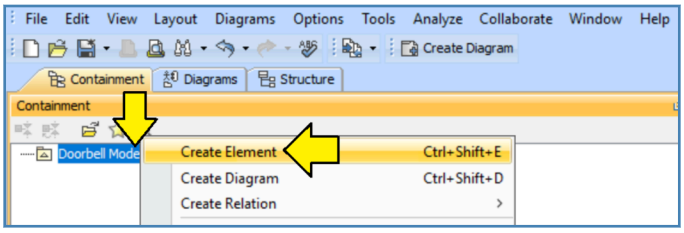


Figure 1-6 – Create the first package

Right-click on the model and select "Create Element" as shown in Figure 1-6.

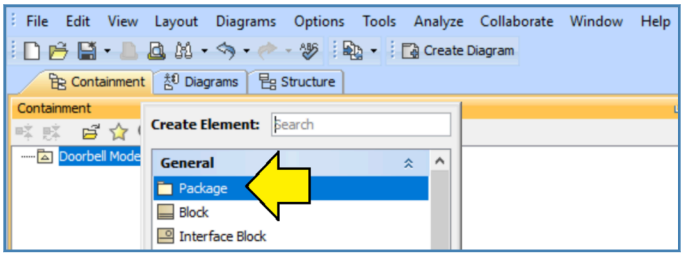


Figure 1-7 – Click on the Package entry

Cameo Systems Modeler creates a package with a default name.

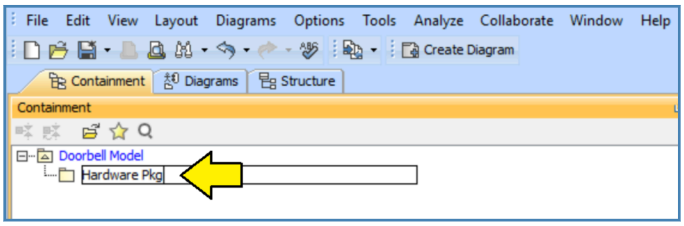


Figure 1-8 – Name the package

Click on the default name and give the package a new name. We recommend that you name the package "Hardware Pkg".

Package Naming Tip

In principle, *Cameo Systems Modeler* really doesn't care how elements are named. As such, it is very easy in the normal course of modeling to end up with a package, an element, and a diagram all named "Car". As your model grows in size, these identical names for different things can get confusing. In order to make the model easier to understand, we recommend getting in the habit of adding "Pkg" to the end of all package names. We will be following this convention for the rest of this book.

Adding a Block Definition Diagram

Now we are ready to add our first diagram. SysML 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.

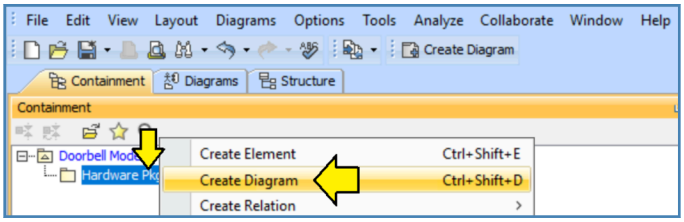


Figure 1-9 – Right-click and add diagram

Right-click on the new package "Hardware Pkg" and select "Create Diagram".

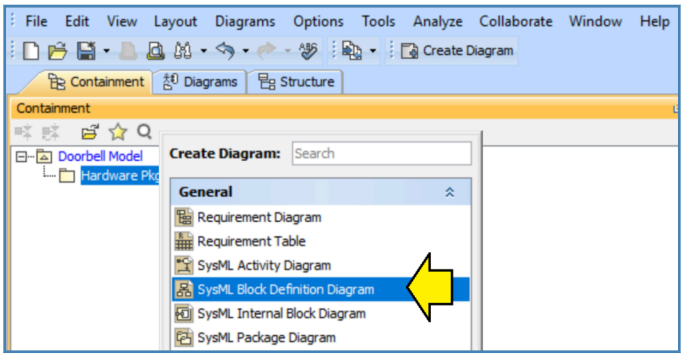


Figure 1-10 – Select SysML Block Definition Diagram

Select "SysML Block Definition Diagram".

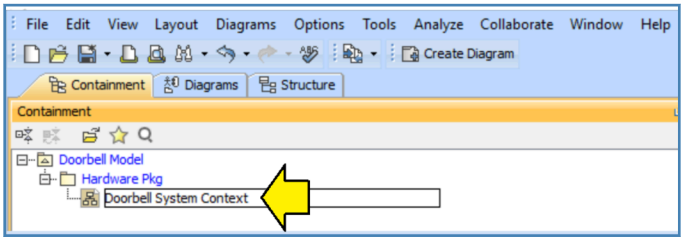


Figure 1-11 – Name the block definition diagram

Name the diagram "Doorbell System Context".

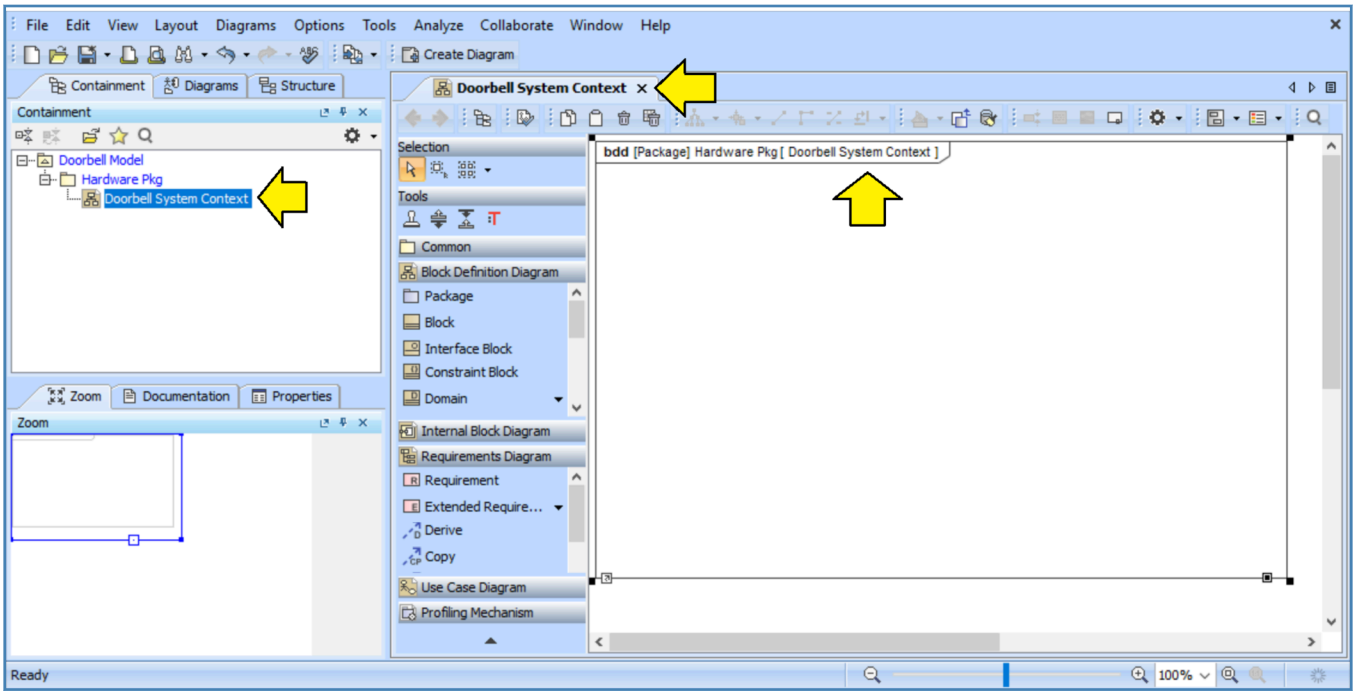


Figure 1-12 – Quick look at the tool

Let's take a quick look at the tool now that we have created the block diagram as shown in Figure 1-12:

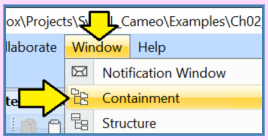
1) On the left side of the screen, we can see a pane with three tabs labeled: "Containment", "Diagrams", and "Structure".
2) This set of three tabs is called the "Model Browser".
3) In this book, we are going to work primarily with the pane called "Containment". *Cameo Systems Modeler* documentation refers to this pane as the **containment tab**.
4) The containment tab presents a hierarchical view of the model that can be expanded and explored similar to the file explorer user interfaces in Windows and other operating systems. This tree is the direct representation of the model. The tree itself (as opposed to the tool pane that contains the tree) referred to as the **containment tree**. In this book, mostly we will be talking about this model tree, not about the pane containing the tree.
5) In the containment tree, we see the new block definition diagram called "Doorbell System Context" in the package "Hardware Pkg".
6) Below the containment tree, we see a pane called "Zoom". When you are working with a very large diagram, this tool can be used to resize and move the view around in the diagram.
7) In the center of the screen, we can see a pane that contains icons for different SysML elements that are appropriate for different types of SysML diagrams. This pane is referred to as the **diagram palette**.
8) To the right of the screen, we see that the tool has created our block definition diagram.

9) 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 top of the screen.

10) Our diagram has a standard SysML frame with a header. In the header, "bdd" stands for "block definition diagram". This diagram lives in the package "Hardware Pkg". The name of the diagram is: "Doorbell System Context".

What if the containment tab isn't there?

What if the containment tree isn't visible on your screen? This kind of thing happens all the time with this sort of highly configurable tool.



The containment tree can be displayed by selecting "Window" and then selecting "Containment".

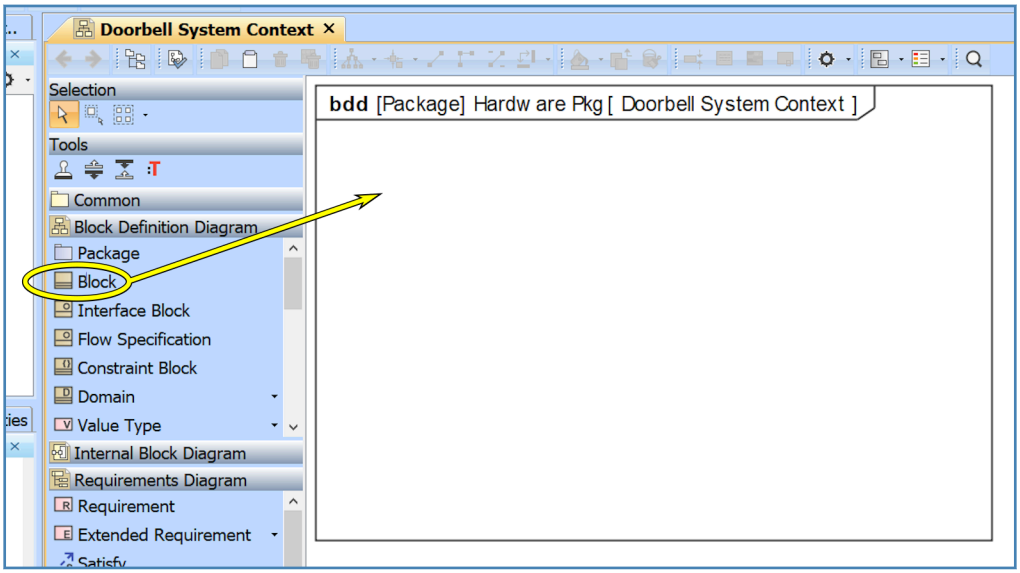


Figure 1-13 – Click on Block icon and drag to diagram

Now we are ready to add our first **block** to our model. In the diagram palette, click on the "Block" icon and drag it to the right into the frame of the block definition diagram as shown in Figure 1-13.

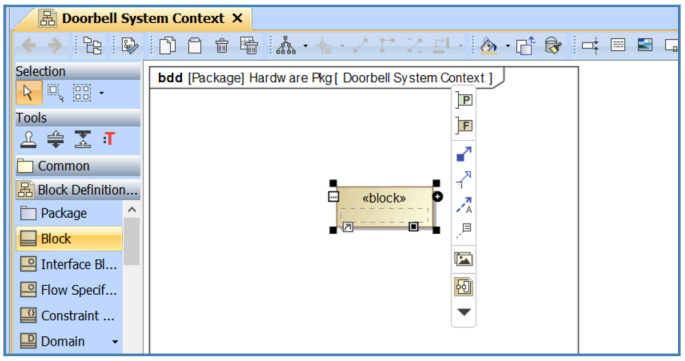


Figure 1-14 – A block will appear in the diagram

A block will appear in the diagram. A toolbar will also appear to the right of the block to allow immediate transition to a next step such as adding a port to the block. You can ignore the toolbar for the moment. (2)

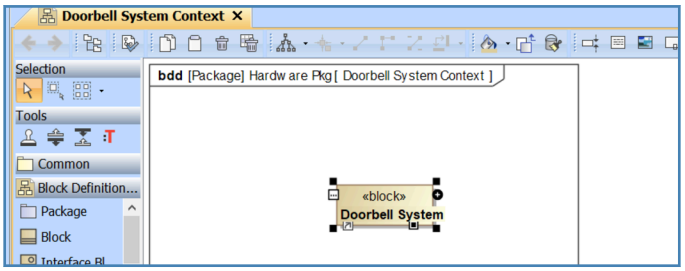


Figure 1-15 – Double-click, name the block, press OK

Just begin typing. The toolbar will disappear. Name the block "Doorbell System". When you are done, click on the diagram outside the block and the block will resize itself to fit the name.

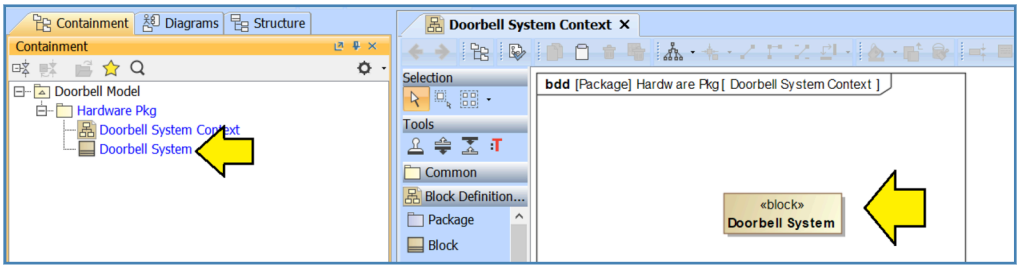


Figure 1-16 – The new block appears in model at left and in the diagram

Examining Figure 1-16 we see the block in two places. We see the block in the containment tree in the left pane. We also see the block in the diagram in the right pane.

(2) This is the *smart manipulator* floating toolbar, see: https://url4ap.net/Cameo-Floating

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 containment tree 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 *actual system* and the diagrams are just *views* of the model. That is, if we think of the model as a building, the diagrams are like snapshots taken of the building from different angles. The snapshots (diagrams) are not the building. The model is the building.

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 containment tree 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.

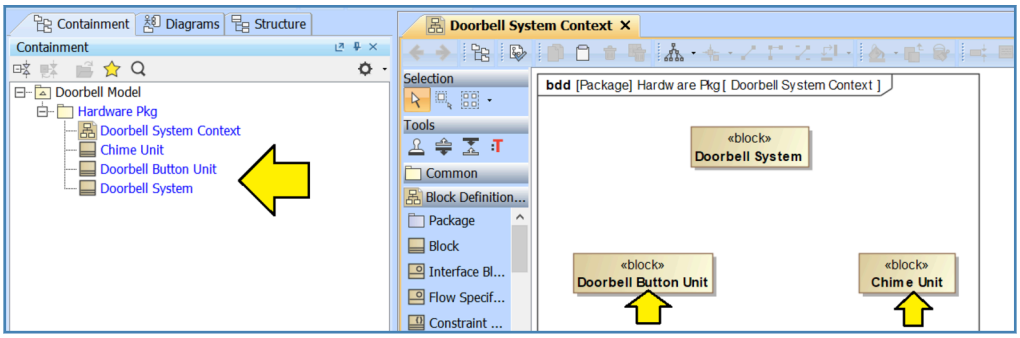


Figure 1-17 – Add two more blocks to the diagram

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** often referred to more simply as **composition**.

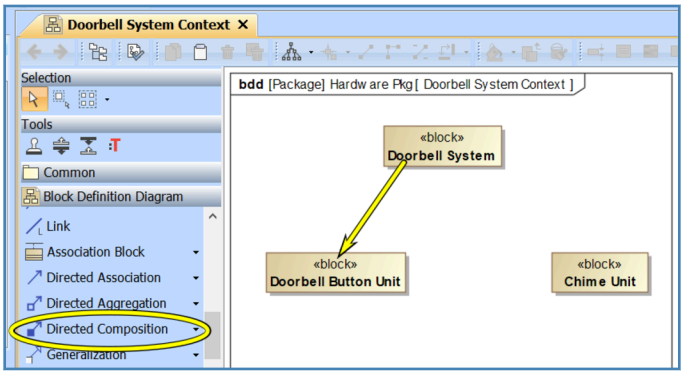


Figure 1-18 – Select the composition relationship and drag as shown

In the diagram palette, use the scrollbar of the section called: "Block Definition Diagram" to scroll down until you find an icon for "Directed Composition". Select this icon. Then drag from the "Doorbell System" block to the "Doorbell Button Unit" block as shown in Figure 1-18.

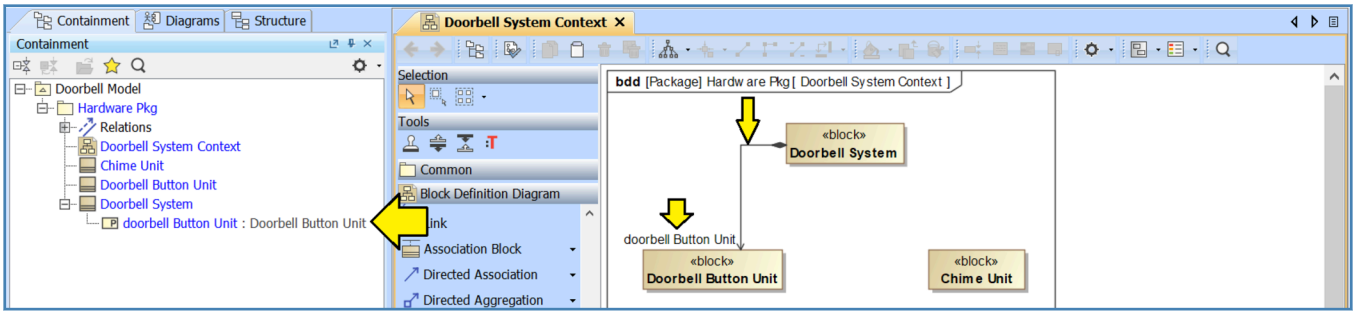


Figure 1-19 – Composition relationship

After you have completed the drag operation to connect the two blocks, three changes will be visible in the tool as shown in Figure 1-19.

1) There is now an arrow from the "Doorbell System" block to the "Doorbell Button Unit" block.
   - At the base of the arrow (the "Doorbell System" block) the connector is a black diamond. The black diamond indicates composition as in: "The Doorbell System is composed of a Doorbell Button Unit."
   - The other end of the arrow is an open arrowhead. This arrowhead indicates the direction of navigation, which does not matter here, but which we will cover a little more in the chapter on block definition diagrams.
2) There is text label next to the open arrowhead "doorbell Button Unit" next to the open arrowhead which we will look at a bit more closely in a minute.
3) In the containment tree, we can see that "Doorbell System" now has something nested underneath it called "doorbell Button Unit:Doorbell Button Unit".

What we have done is modeled the idea that the "Doorbell System" contains a **part**. The tool has given the part a provisional name of "doorbell Button Unit" – that is, it has taken the block name, made the first character lower case, and assigned that as the part name.

What is going on here is that the definition of a block is actually a "type" declaration and the declaration of a composition relationship actually defines an "instance" of the associated block (type). However, we are not going to worry too much about the differences between "types" and "instances" for the moment. We will cover that topic in more detail in later chapters.

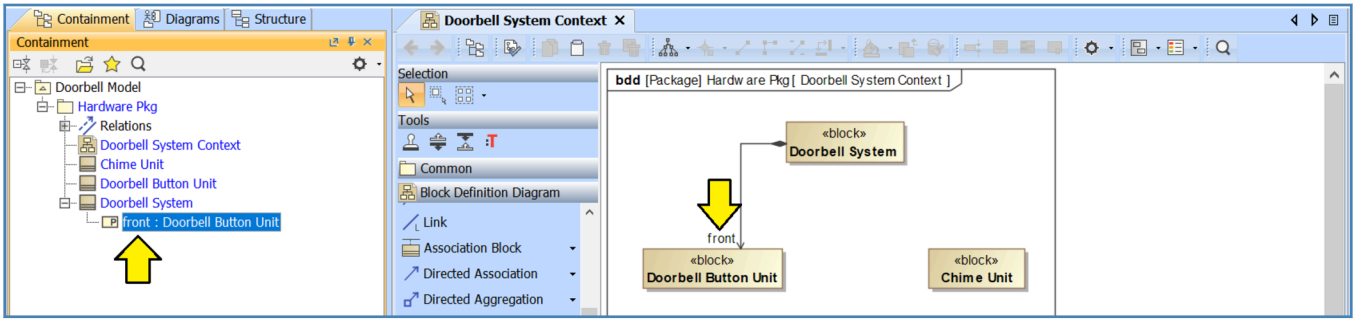


Figure 1-20 – Name the part

Before we continue, let's give the part a more useful name. In the containment tree, 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. [3]

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

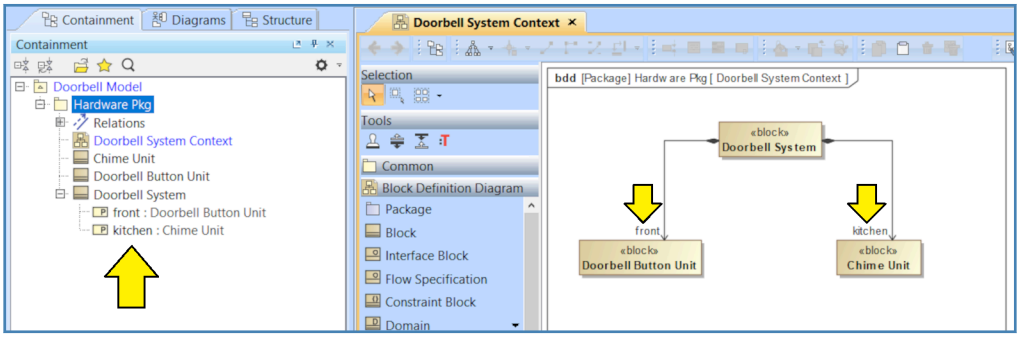


Figure 1-21 – Both part relationships

Your diagram should now look like Figure 1-21.

[3] 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.

Next, we will add an **actor** called "Visitor" to the diagram. In the diagram palette, close the section "Block Definition Diagram" and scroll down to the section for "Use Case Diagram". [4] In that section, you will find an icon for "Actor".

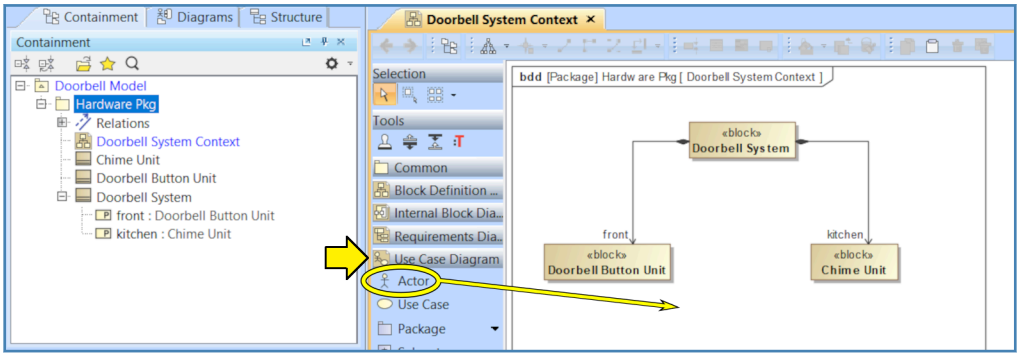


Figure 1-22 – Drag the actor icon to the diagram

Drag the "Actor" icon to the diagram.

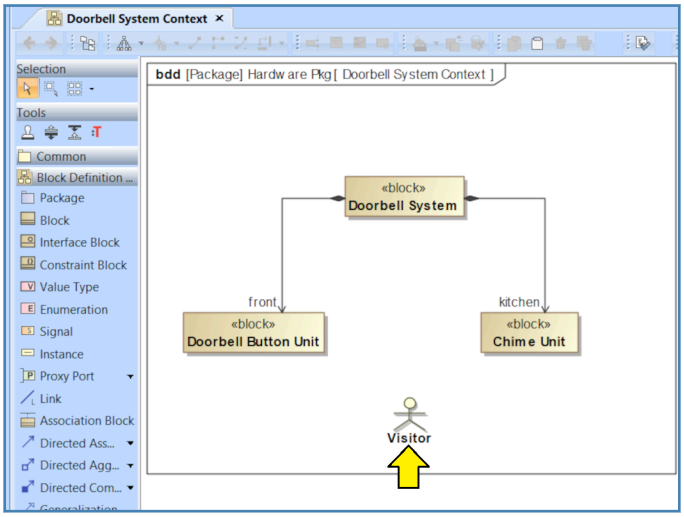


Figure 1-23 – Name the actor

Name the actor "Visitor".

[4] You may need to click on the little black triangle below the "Block Definition Diagram" bar to expand the "expert" options.

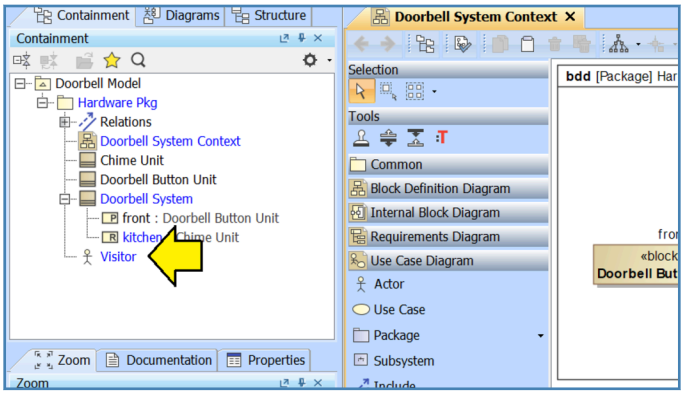


Figure 1-24 – Visitor now visible in containment tree

We can now see that the visitor is visible in the containment tree. However, since our actor is a human [5], we will not want to model the actor as belonging to the package "Hardware Pkg". Add a new package to the model called "Actors Pkg". (See the procedure previously outlined in *Adding a Package* on page 3.)

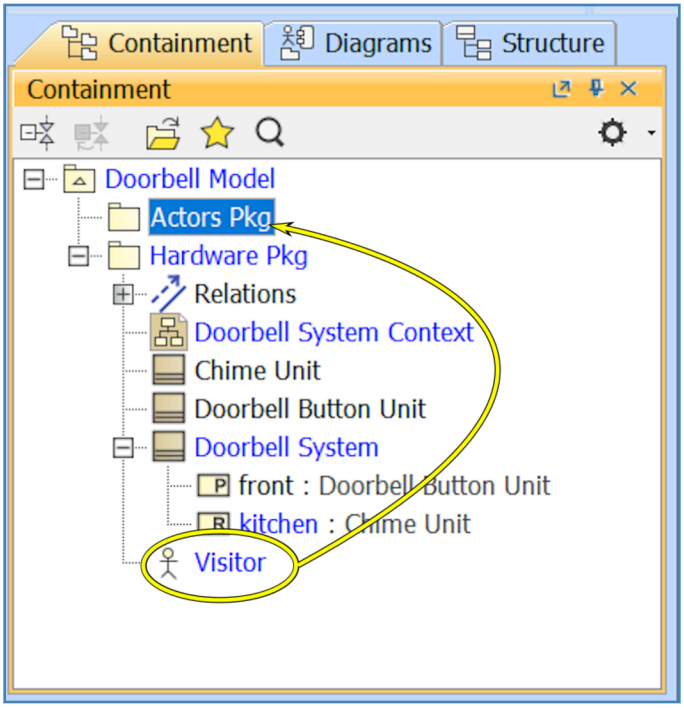


Figure 1-25 – Drag visitor to the Actors package

Drag the visitor from the "Hardware Pkg" package to the "Actors Pkg" package.

[5] 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.

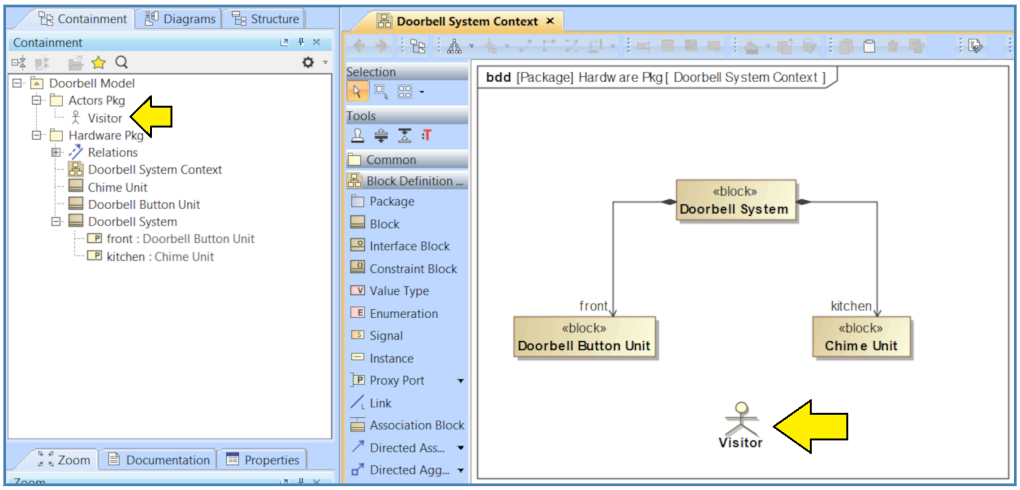


Figure 1-26 – Visitor now in proper package in containment tree

Your diagram should now look like Figure 1-26. Notice 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 is an important point: 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.

System Context Diagram

SysML does not actually define something called a "System Context Diagram". However, a block definition diagram can be used to model the system context. Nailing down the system context is an important early step in any disciplined systems engineering effort. Specifically, a skilled systems engineering practitioner will help the stakeholders nail down the definition of which things are part of the system being designed versus which things are merely "nearby".

This activity can be surprisingly difficult! Typically, the meeting will start with the stakeholders looking bored and a little irritated. Some will be asking why they have to be at this meeting. Others will be confidently asserting that "Everyone knows what is in the system anyway." However, as soon as the systems engineering practitioner starts drawing the specifics on the whiteboard, the room will explode into fractious disagreement. In most cases, the stakeholders will have overestimated their level of common understanding and consensus.

Adding a Requirement Diagram

Now we are ready to add a few requirements to our system. In order to work with requirements, we will use a **requirement diagram**. The first thing we will want to do is add a package to the model called: "Requirements Pkg" (See the procedure previously outlined in *Adding a Package* on page 3.)

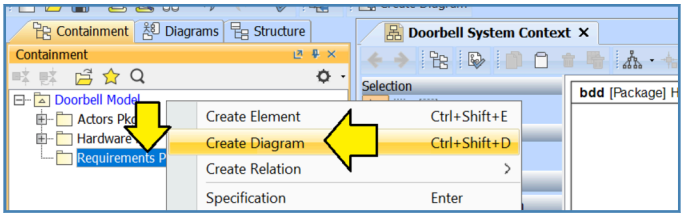


Figure 1-27 – Right-click and select Create Diagram

Right-click on the "Requirements Pkg" package and select "Create Diagram".

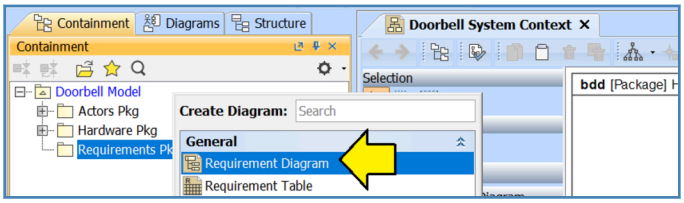


Figure 1-28 – Select Requirement Diagram

Select "Requirement Diagram".

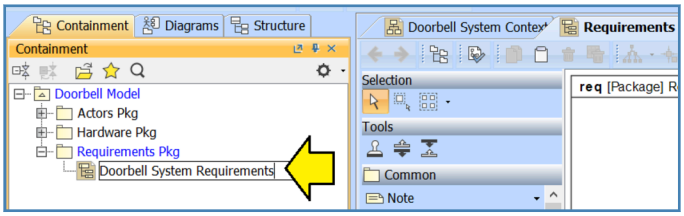
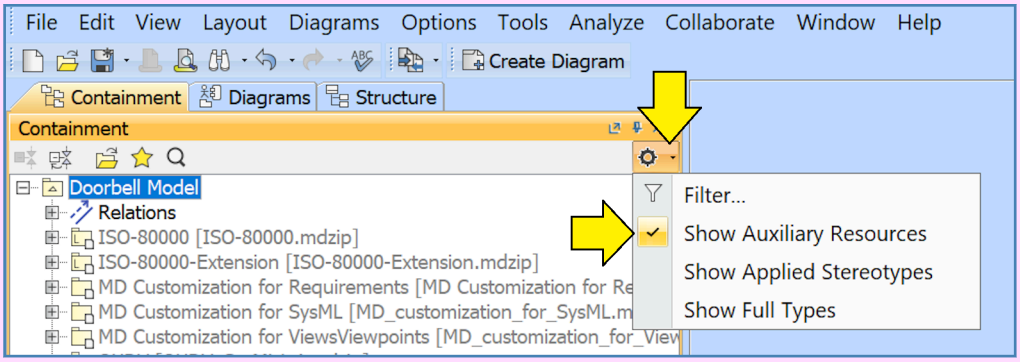


Figure 1-29 – Name the diagram

Name the new diagram "Doorbell System Requirements".

Hiding Auxiliary Resources

Depending on how your copy of the tool is configured, you may see what looks like 8-10 "ghost packages" with names like "ISO-8000-Extension". For the most part, we will not be making use of these in this book and they can be hidden.



In the upper right-hand corner of the containment tree, you will find a gear icon. Pull down the menu from this icon and unselect "Show Auxiliary Resources".

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

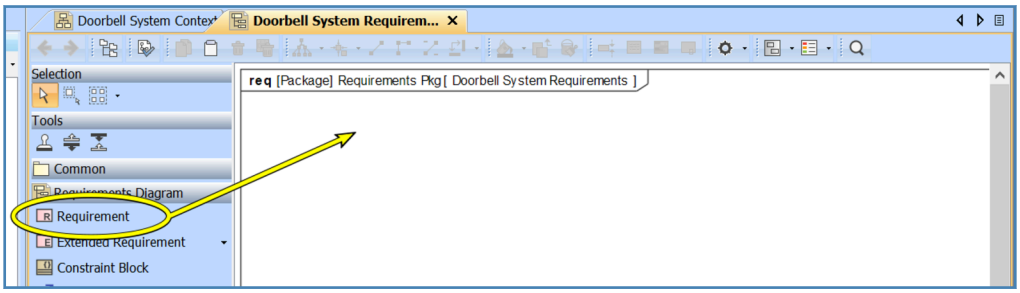


Figure 1-30 – Drag the requirement icon from the Toolbox to the diagram

Drag the requirement icon from the diagram palette to the requirement diagram.

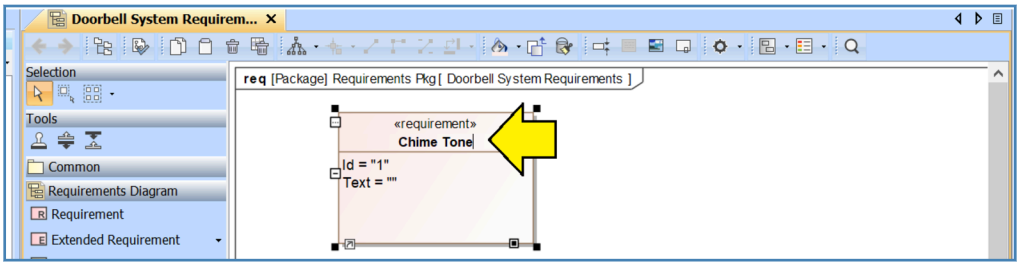


Figure 1-31 – Give the requirement a name

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 want to do is set the **id** and **text** for the requirement. SysML is built on top of UML and UML considers these to be "tags" for the requirement element. *Cameo Systems Modeler* supports two slightly different methods for setting the value of a tag. We will use one method to set the value of the id and the other method to set the value of the text.

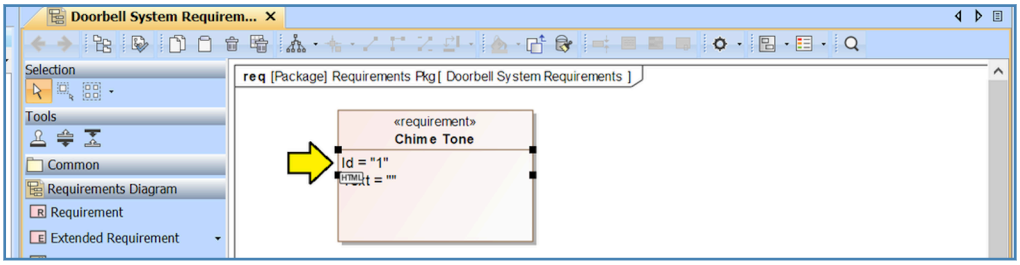


Figure 1-32 – Single-Click method: click once to select Id tag

First, we will use the single-click method. Click once to select the "Id" tag as shown in Figure: 1-32. [6] Notice that the "Id" tag is now surrounded by four small black boxes to indicated that it has been selected.

[6] Sharp-eyed readers may notice that we are using both "Id" and "id" in our explanation here. The SysML standard itself is somewhat confused on this point, using both. The preponderance of the standard seems to be "id". However, the diagram example in the standard shows "Id". *Cameo Systems Modeler* seems to be following the one diagram example in the standard. As authors, we are navigating the small inconsistencies in the tools and the standard as best we can.

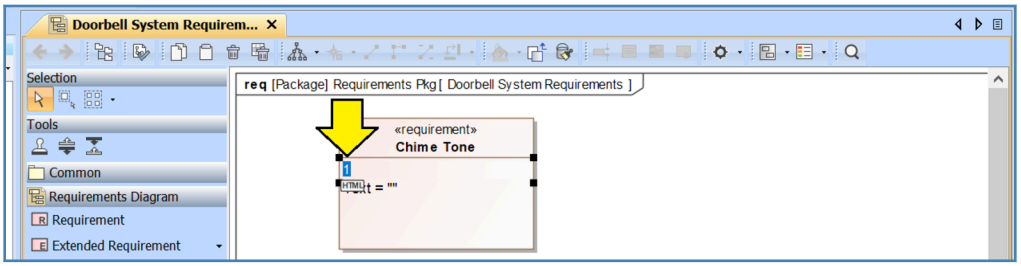


Figure 1-33 – Single-Click method: click again to enter edit mode

Click once again to enter edit mode.

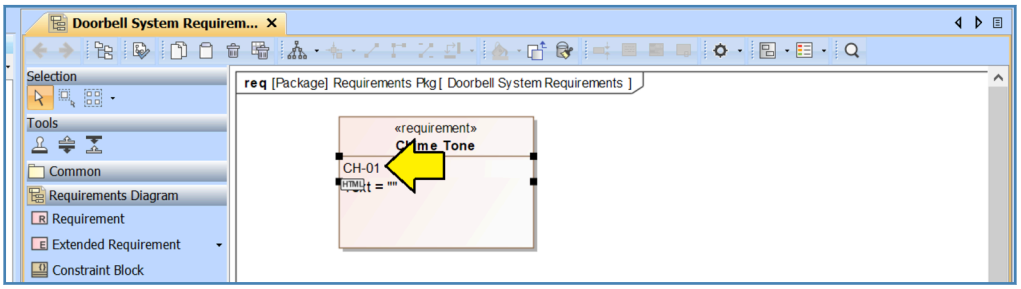


Figure 1-34 – Single-Click method: click again to enter edit mode

Set the id. 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". When you are done, click anywhere on the diagram outside of the requirement element to let the tool know that you are done editing the requirement id.

Next, we will use the double-click method to set the text of the requirement.

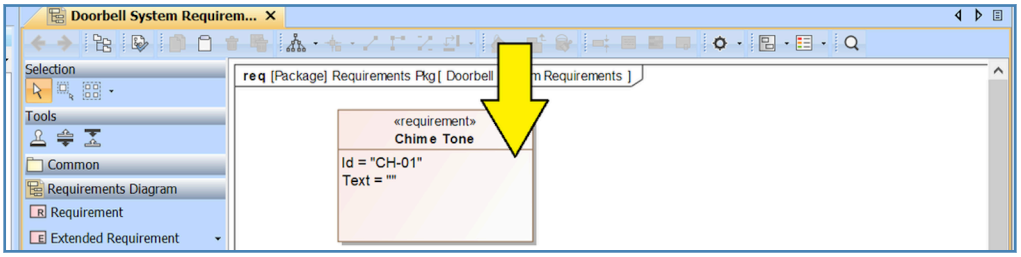


Figure 1-35 – Double-click on the requirement

Double-click on the requirement in the diagram.

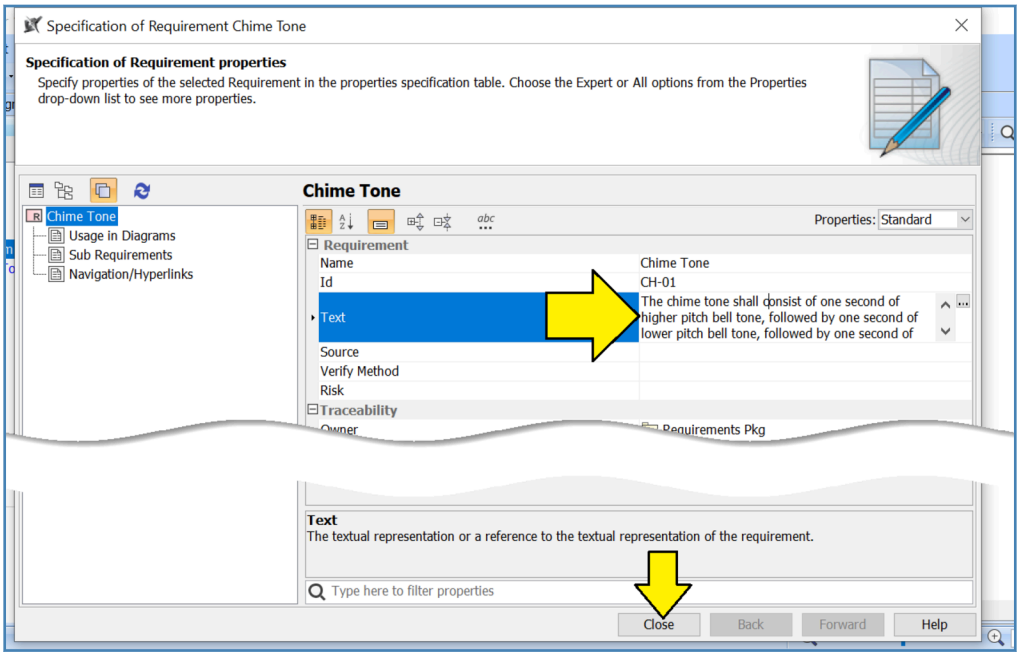


Figure 1-36 – Enter the text of the requirement

A dialog box will appear. Click in the box to the right of "Text". 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 "Close". [7]

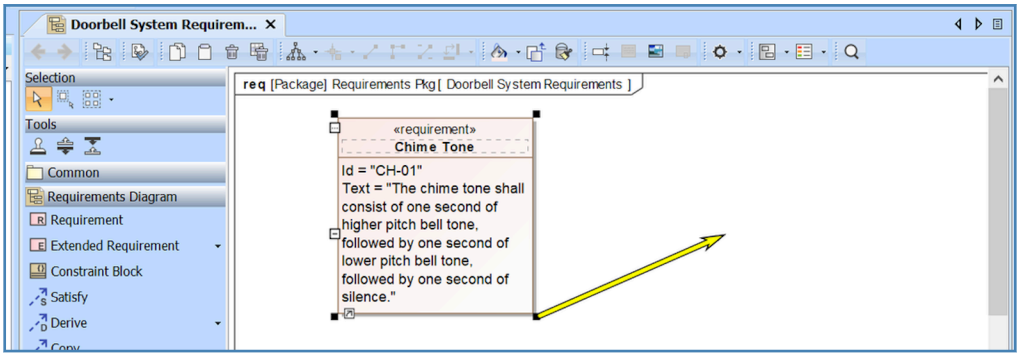


Figure 1-37 – Drag the corner of the requirement box to resize it

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. Click once on the requirement. Four black boxes will appear, one in each corner. After that, you can simply grab one of

[7] At the top right of this dialog box, you will notice a field called "Properties" set to "standard". If this dialog box seems to contain many more strange fields than shown, your copy of the tool may be set to "expert" or "all" in which case you might want to set it back to "standard" to reduce the number of choices.

the four black box icons with the mouse and drag to resize the requirement. 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
>
> Ctrl+W is also very useful. Ctrl+W will scale the diagram to fit the contents to the space in the pane.

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 marketing department has identified this weakness in traditional doorbell design as an opportunity for product differentiation. The marketing department has developed this brief **user story** to describe the desired behavior for the doorbell system:

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. [8]

Starting with marketing's user story, our requirements engineering team has developed the following small set of **requirements** for our doorbell system:

[8] The observant reader will notice that we have just created the beginnings of a "user story" for our system. User stories (sometimes called "scenarios") are an important part of almost any design methodology. Design methodologies are mostly beyond the scope of this book. However, there is a section on methodology in the *Challenges Ahead* chapter of this book. There are also several good methodology books in the *Further Reading* appendix which is also available from the Asatte Press website here: https://url4ap.net/Reading

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 1-38 – Requirements for the doorbell system

Go ahead and repeat the procedure and add the remaining five requirements to the model. In order to save you time and effort in typing, we have provided a spreadsheet of these requirements in the example files in the directory *Examples\Ch02_Quick_Start\S2.5_Req_Diag*.

Importing from a Spreadsheet: Generic Table Wizard

Although the setup is a little too intricate to cover here, *Cameo Systems Modeler* supports a very useful feature known as the "generic table". If you select "Diagrams" from the top menu bar and then "Diagram Wizards" and "Generic Table Wizard...," you will find a wizard that will help you create a requirements table that looks very similar to the spreadsheet. Once you select the proper tags and drag the columns around to match the order in the spreadsheet, you will be able to directly copy-and-paste content to and from the spreadsheet. In the example file for this section, if you look in the package "Requirements" you will find a table called "Requirements" which is exactly such a table. New requirements can be created in this table directly or copied from new rows in the spreadsheet.

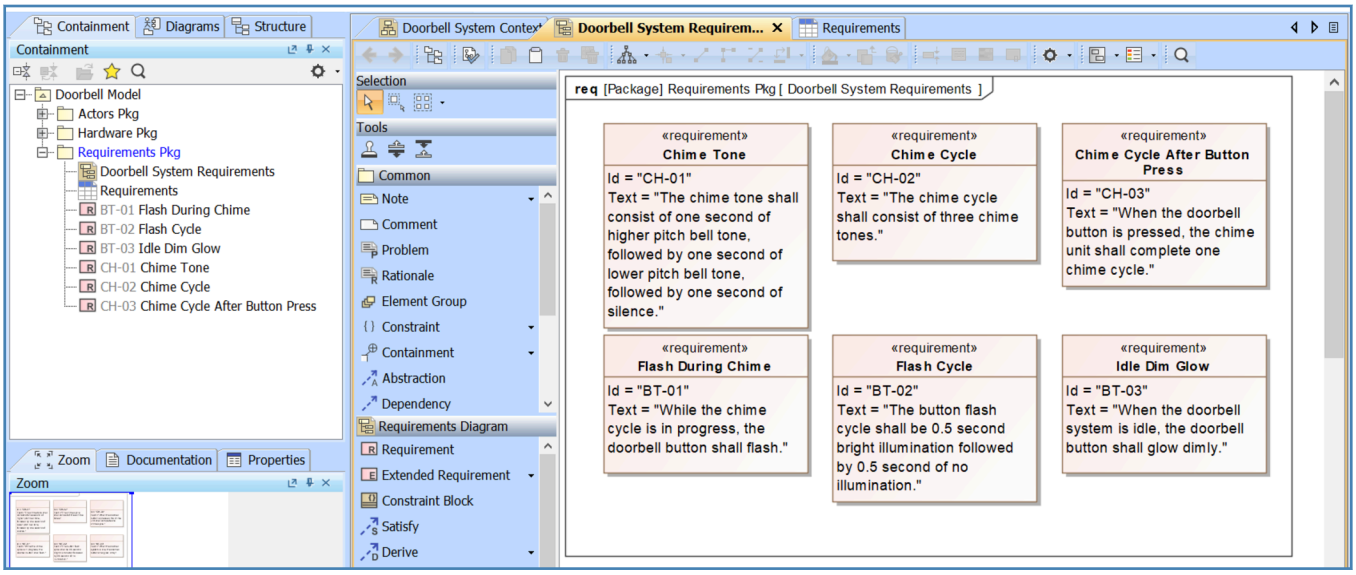


Figure 1-39 – Completed requirements for the doorbell system

When you are done adding the requirements, your requirement diagram should look like Figure 1-39.

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 requirement diagram called "Button Glow Detail".

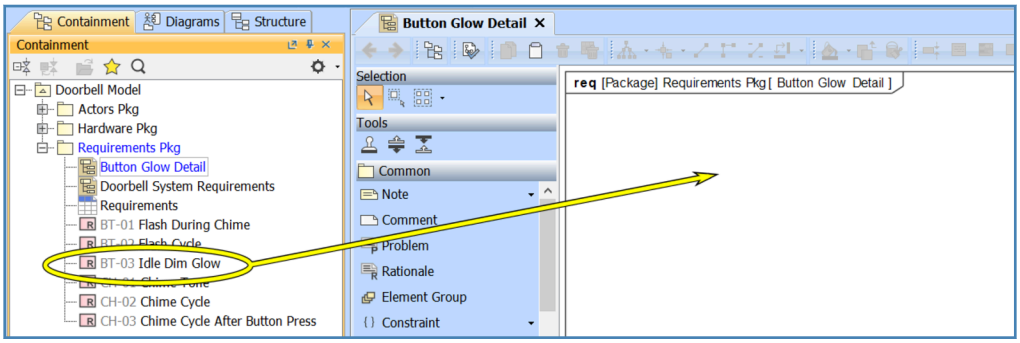


Figure 1-40 – Drag the idle dim glow requirement to the diagram

Drag the "Idle Dim Glow" requirement to the diagram.

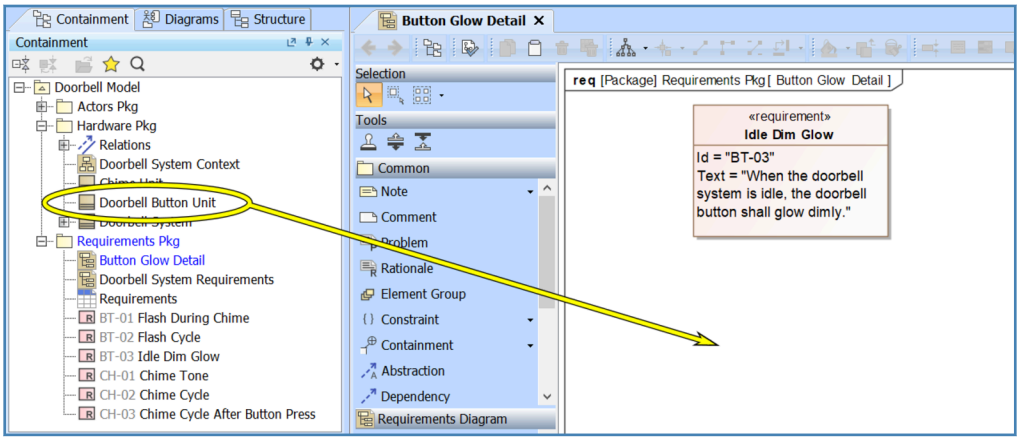


Figure 1-41 – Drag the doorbell button unit block to the diagram

Next drag the "Doorbell Button Unit" block to the diagram.

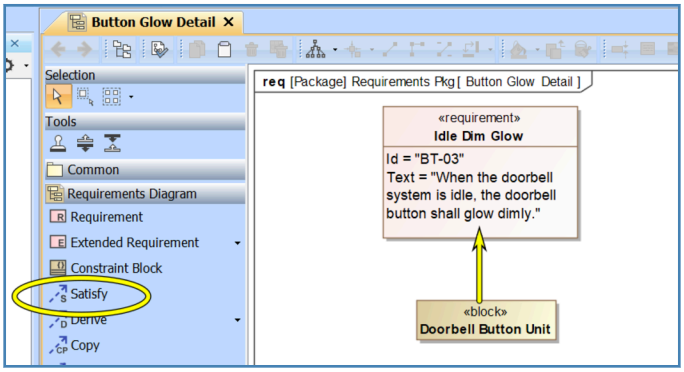


Figure 1-42 – Select satisfy and drag relationship

In the diagram palette select the "Satisfy" icon and drag a **satisfy** relationship from the button unit to the requirement.

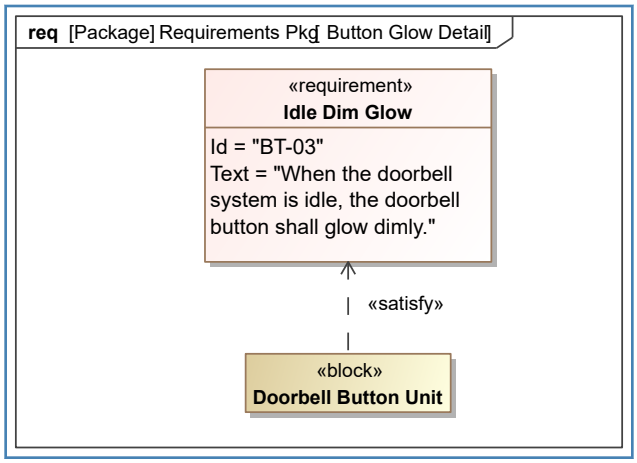


Figure 1-43 – Completed satisfy relationship

When you are done adding the satisfy relationship, your requirement diagram should look like Figure 1-43. This diagram can be understood as meaning: "The doorbell button unit is responsible for satisfying the idle dim glow requirement".

Quickly examining the diagram palette, the reader will notice that there are several more descriptive relationships. In the chapter on requirement 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 1-43, 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 four behavioral diagrams:

- Activity diagram
- Sequence diagram
- State machine diagram

- Use case diagram

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

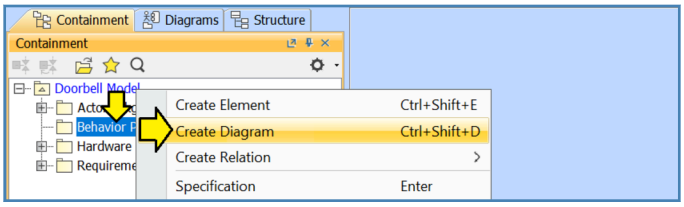


Figure 1-44 – Add package and diagram

Add a new package for "Behavior Pkg" to the model. Right-click on the new package in the containment tree and select "Create Diagram".

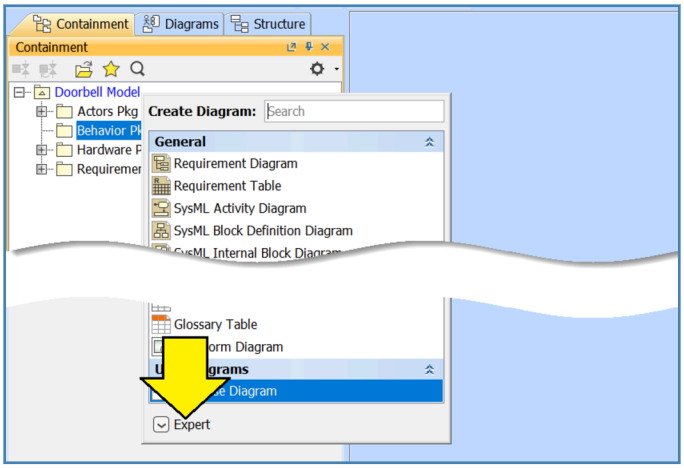


Figure 1-45 – Click on Expert

For some reason, in the "Create Diagram" dialog *Cameo Systems Modeler* shows only eight of the nine standard SysML diagram types. The sequence diagram is the one type not displayed. In order to gain access to the sequence diagram, click on the "Expert" control at the bottom of the dialog box.

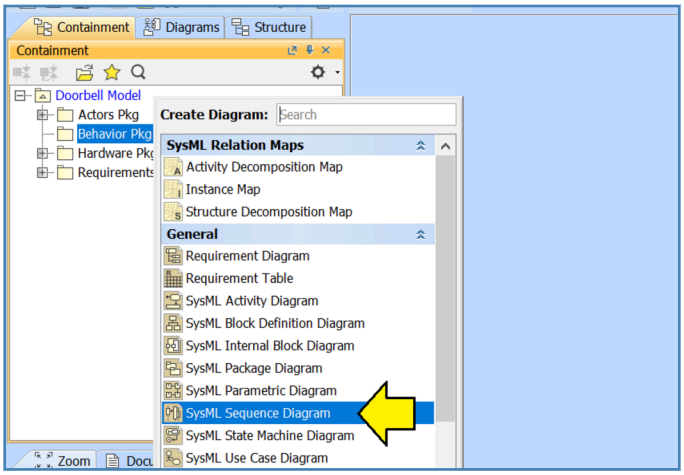


Figure 1-46 – Select sequence diagram

The SysML sequence diagram will be added to the list of available diagram types. Select "SysML Sequence Diagram".

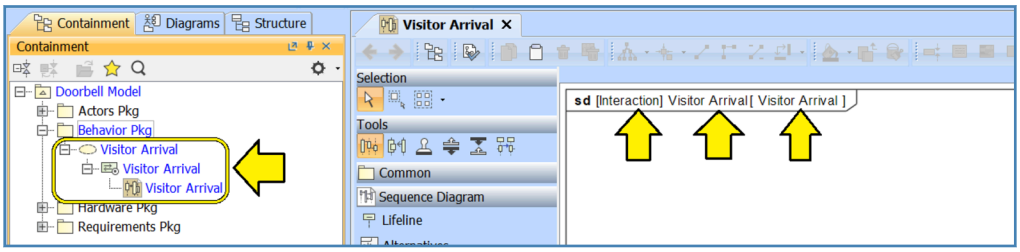


Figure 1-47 – Name and create diagram

Name this diagram: "Visitor Arrival".

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.

Actually, what *Cameo Systems Modeler* has done here is create three things with the same name:

1) A UML **collaboration** element
2) Inside the collaboration element, an **interaction** element
3) Inside the interaction element, a sequence diagram

In UML, the collaboration is a collection of instances that have roles in the collaboration and connect to each other. Don't worry too much about the collaboration for the moment.

The interaction represents the model of how the instances in the collaboration interact with each other.

In UML there are four different kinds of diagrams to represent an interaction. The SysML committee decided to retain only the sequence diagram. *Cameo Systems Modeler* is actually a UML tool running a SysML profile for us. As such, it creates both a collaboration and an interaction when we ask it to create a sequence diagram.

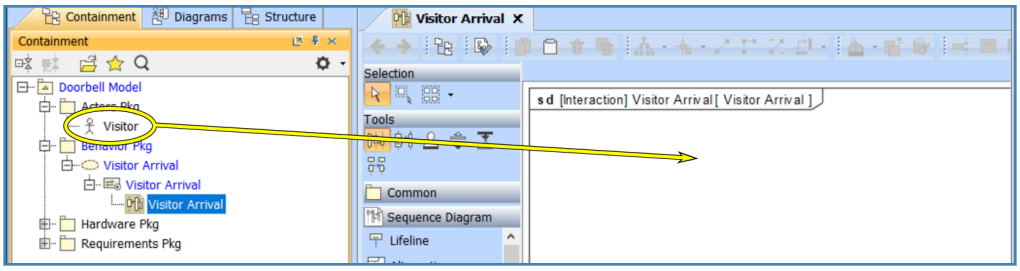


Figure 1-48 – Drag the visitor to the diagram

Drag the visitor from the containment tree to the diagram.

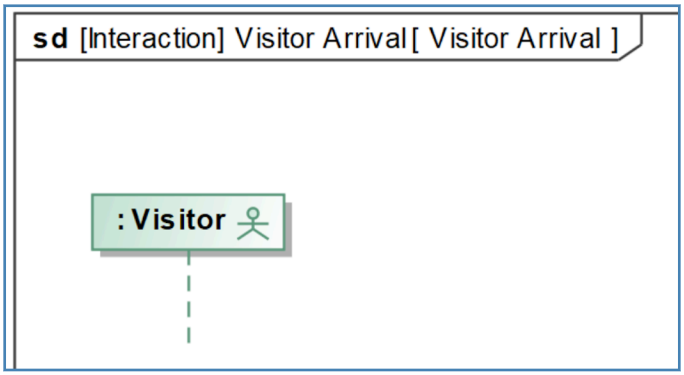


Figure 1-49 – Lifeline

Cameo Systems Modeler will create something called a **lifeline** for the visitor element.

- At the top is the element itself.
- 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.

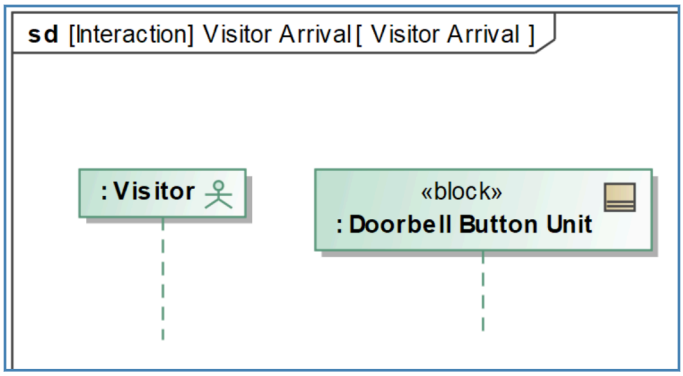


Figure 1-50 – Drag the doorbell button unit to the diagram

Next drag the block named "Doorbell Button Unit" from the containment tree to the diagram.

Drag the block named "Chime Unit" to the diagram as well.

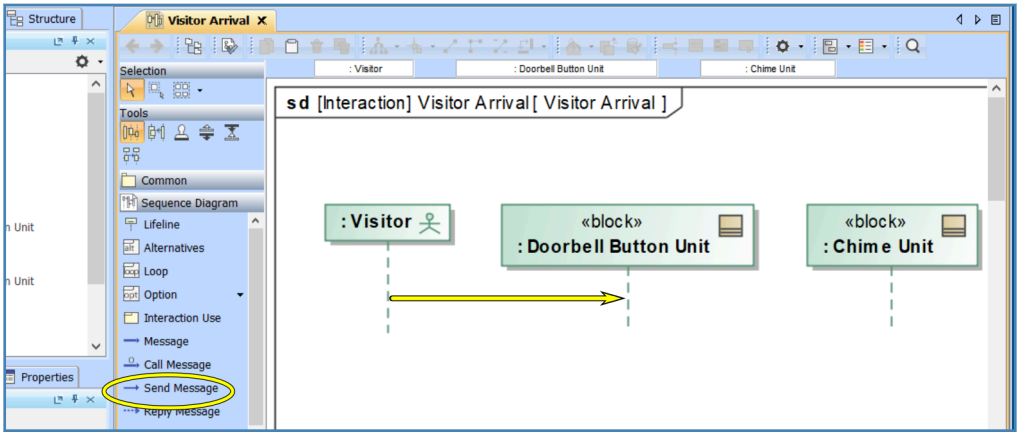


Figure 1-51 – Drag message from visitor to button

We now have three lifelines for three elements. Now we are ready to start creating the actual interaction between these three elements. In the diagram palette select "Send Message" and drag from the lifeline for the visitor to the lifeline for the doorbell button unit. Look for the dashed lifeline to turn blue as you start the dragging motion on the visitor's lifeline and finish it on the doorbell button unit's lifeline.

If you release the mouse button when the target lifeline is not in the blue state, the tool will create a new lifeline which can be confusing. If you accidentally create such an unwanted lifeline, press control-Z to back up one step and try again.

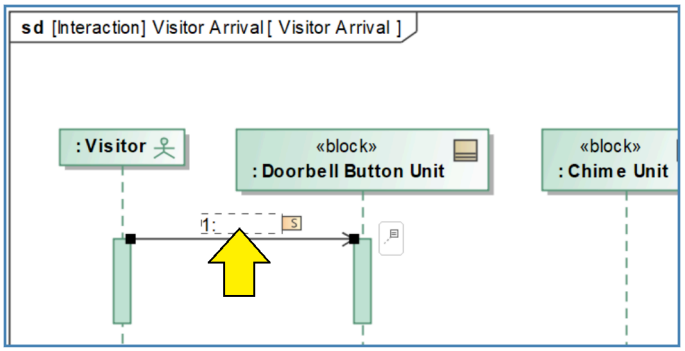


Figure 1-52 – Name the message

We now have our first **message** on the diagram. After you release to create the message, the tool will present you with an empty window just above the message arrow. This box is for the message name. Name the message: "Push Button".

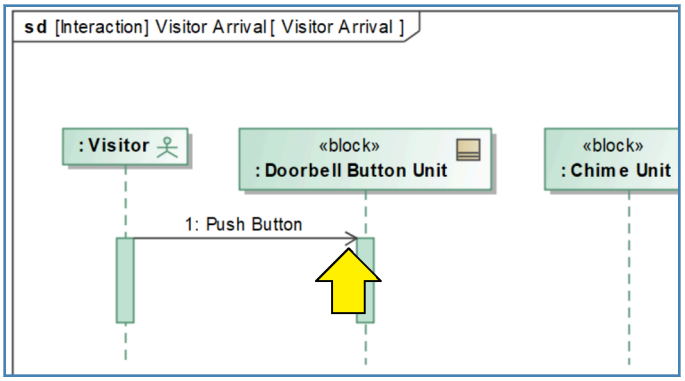


Figure 1-53 – The message is asynchronous

If you look carefully at the arrow created by the tool, the arrowhead is open. This open arrowhead indicates an asynchronous message in SysML. 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. However, if we did want a synchronous message (shown in SysML with a solid arrowhead) we could use the "Call Message" control instead of the "Send Message" control to create the message arrow.

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.

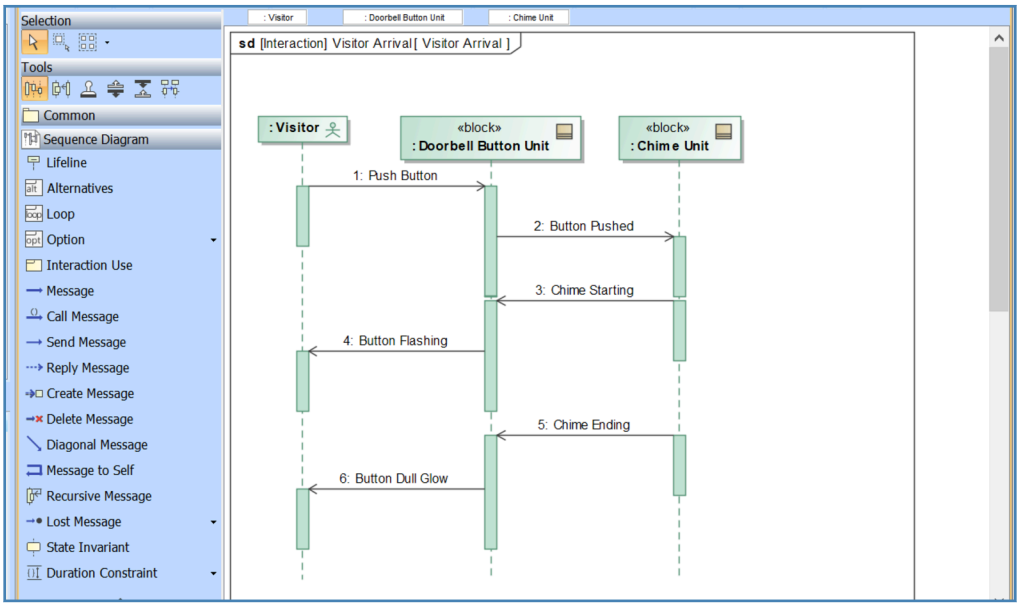


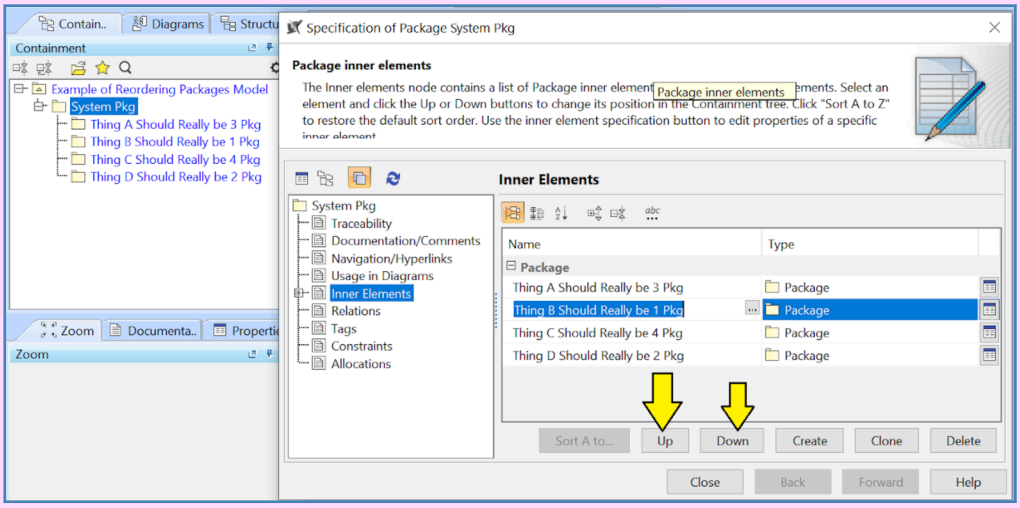
Figure 1-54 – Finished sequence diagram

Your finished sequence diagram should look like Figure 1-54.

This concludes our quick look at SysML and at the *Cameo Systems Modeler* 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.

Ordering Things in the Containment Tab

By default, *Cameo Systems Modeler* sorts elements in the containment tree in alphabetical order. What if you want them sorted in some other order?



If you select the parent element, open its specification, and select: "Inner Elements" you can use the up and down arrows to adjust the ordering of contained elements in the containment tree.