Papyrus Tutorial

A Practical Guide to Systems Modeling
with Papyrus and SysML

Version 0.1.3

Contents

List of Figures

1 INTRODUCTION

1.1 Overview

The purpose of this tutorial is to use the *Papyrus* program to create a model of a simple system utilizing the SysML language. A vehicle climate control unit will be modeled starting with the Problem Domain and then moving to the Solution Domain.

1.2 Assumptions

This tutorial assumes that *Papyrus* and the SysML tools have been properly installed and will not cover the procedures to do so. This is not a SysML tutorial but rather how to model an example system with SysML in the Papyrus program. This tutorial uses *Papyrus* version 2019-03 and SysML 1.4 running on a Windows machine. As the tutorial progresses I will assume that you have learned from the previous steps. For example, in the beginning I will describe step-by-step how to create a new package but later in the tutorial I will ask you to create a new package and will assume that you know the procedures in *Papyrus* to do so and they will not be repeated.

2 PROBLEM DOMAIN

2.1 Introduction

In the problem domain we will analyze stakeholder needs and refine them with SysML model elements to gain a clear understanding of the problem that the System Of Interest (SoI) must solve. We will first analyze the SoI from a black-box perspective where we focus on the interaction of the system with the environment without knowledge of the internal structure of the SoI. We then analyze the system from a white-box perspective to perform the functional analysis of the system.

2.2 Black-box perspective

We will start this project by organizing our model and capturing our stakeholder needs.

2.2.1 Step 1. Create the new project and new model.

1. Create a new *Papyrus* project by selecting **File > New > Papyrus Project** (see figure 1).

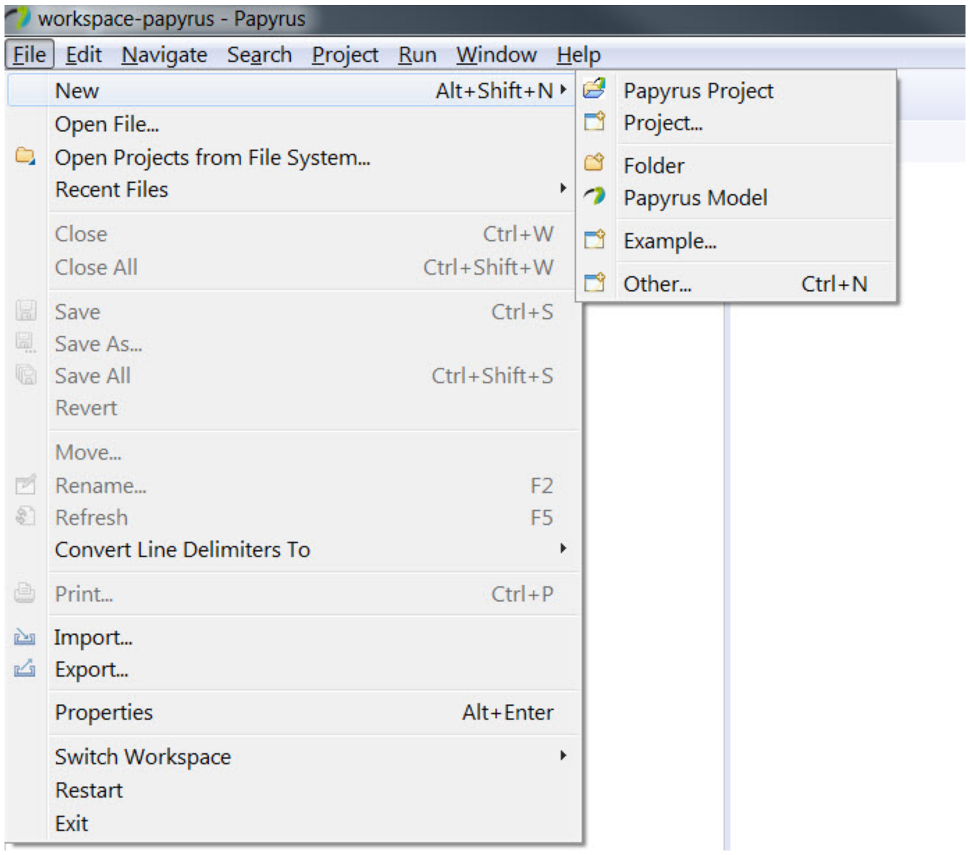
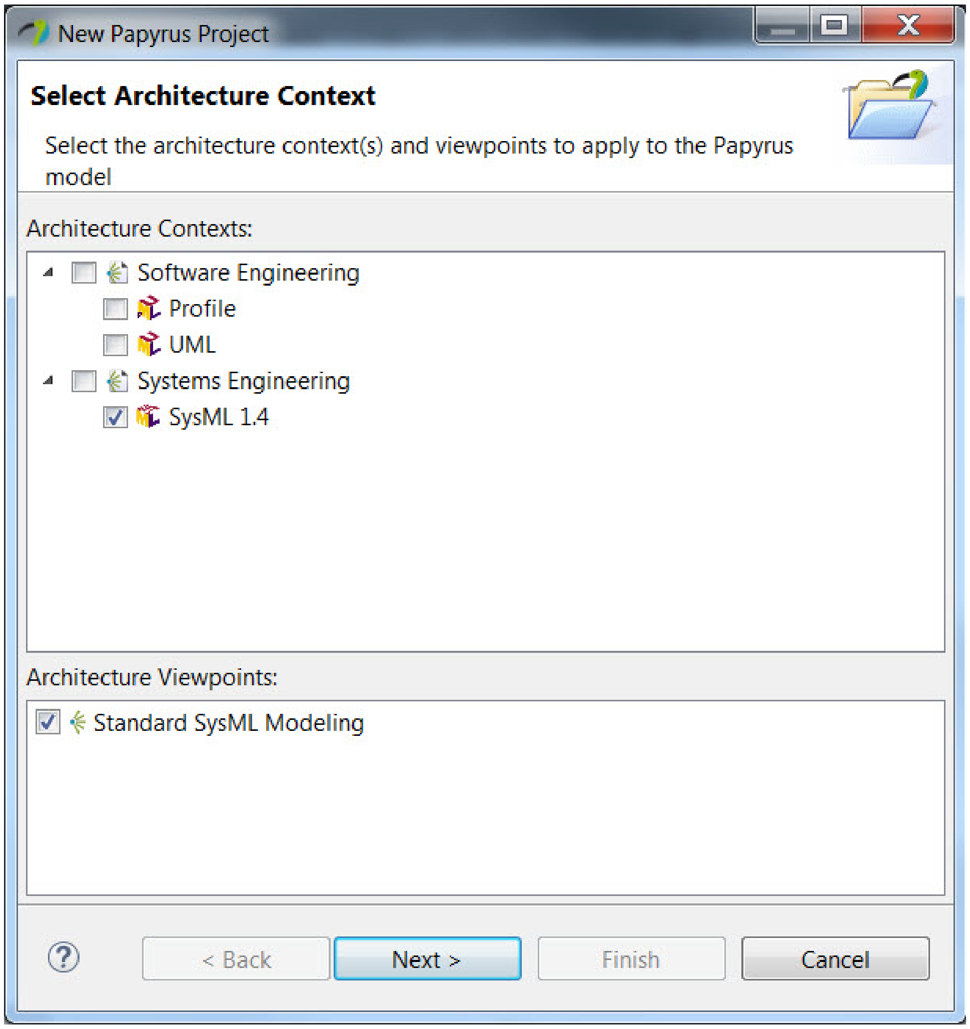


Figure 1: New Papyrus Project

2. The *Select Architecture Context* dialog will then appear. Ensure that the *SysML 1.4* checkbox is selected and then select **Next** (see figure 2).



Select Architecture Context

Select the architecture context(s) and viewpoints to apply to the Papyrus model

Architecture Contexts:

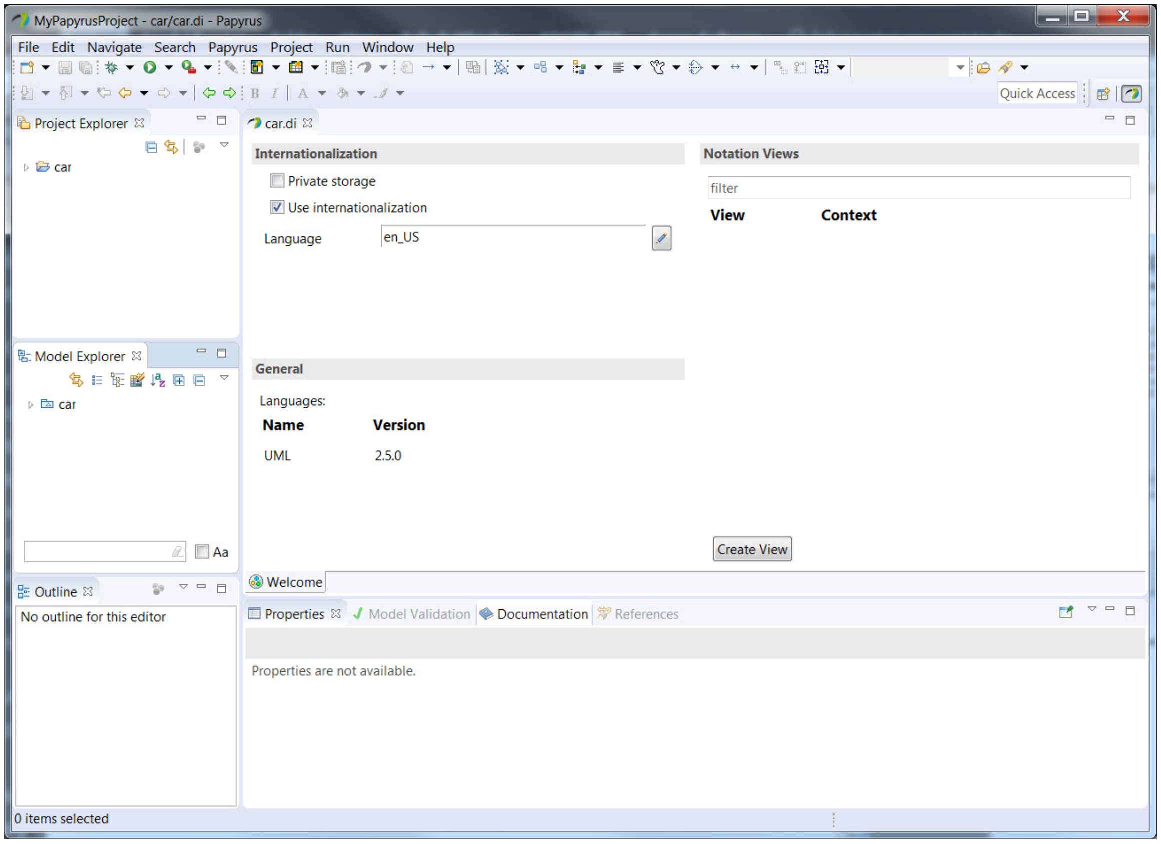
- [ ] Software Engineering
  - [ ] Profile
  - [ ] UML
- [ ] Systems Engineering
  - [✓] SysML 1.4

Architecture Viewpoints:

- [✓] Standard SysML Modeling

Figure 2: Select Architecture Context

3. On the next screen type *car* for the project name and click **Finish**. Your screen should look like figure 3.



Internationalization

- [ ] Private storage
- [✓] Use internationalization

Language: en_US

General

Languages:

Name	Version
UML	2.5.0

Notation Views

filter

View	Context

No outline for this editor

Properties are not available.

Figure 3: Project File Created

4. Now we will create our model for the vehicle climate control system problem space. Create a new *Papyrus* model by selecting **File > New > Papyrus Model** (see figure 4). Ensure that the *SysML 1.4* checkbox is selected when the *Select Architecture Context* dialog appears and then select **Next** (see figure 2).

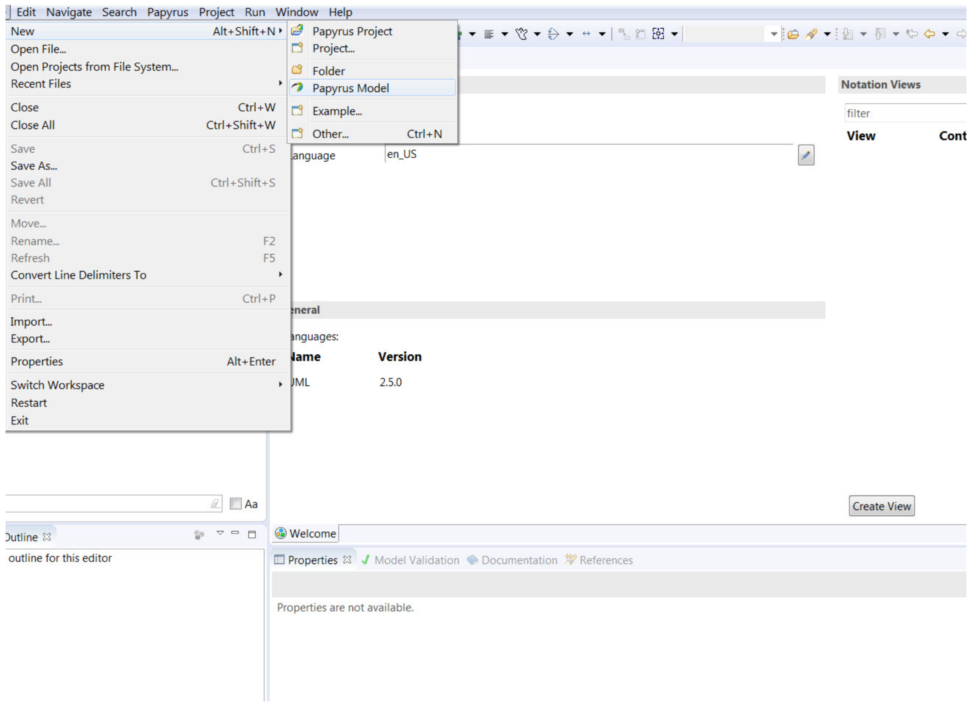
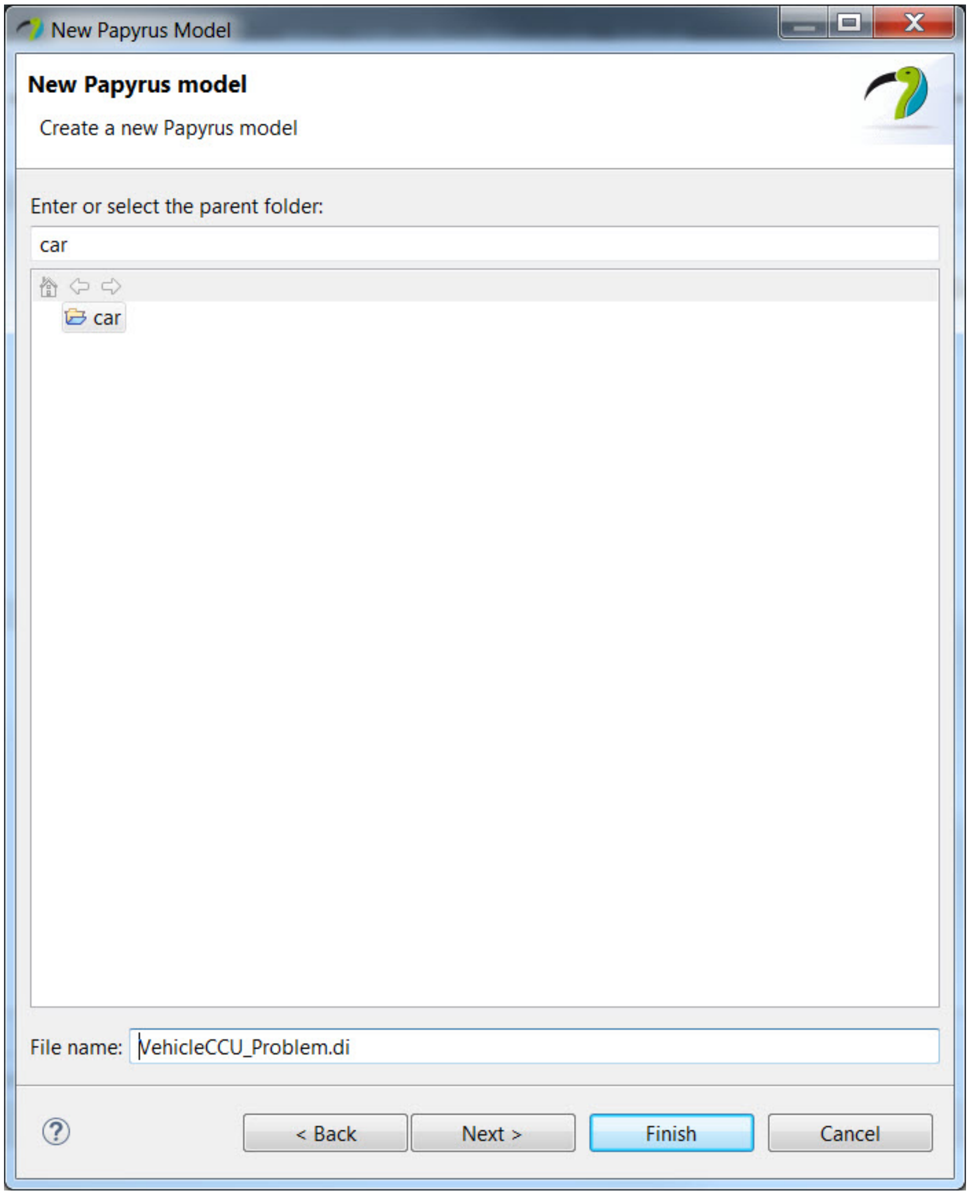


Figure 4: Create New Model

5. When the *New Papyrus Model* dialog appears, click on the *car* folder to choose this as the parent folder and enter *VehicleCCU_Problem* in the file name box and click **Finish** (see figure 5).


New Papyrus Model

New Papyrus model

Create a new Papyrus model

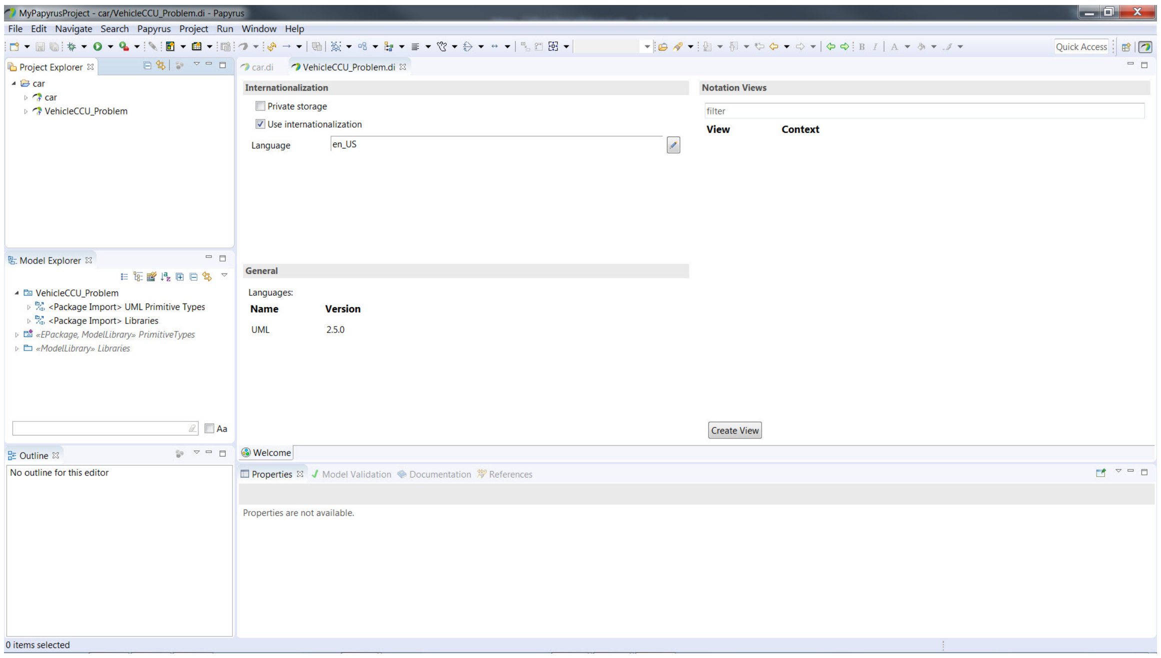
Enter or select the parent folder:

car

File name: VehicleCCU_Problem.di

Figure 5: New Model Dialog

6. Your screen should now look like figure 6. You see that in the *Project Explorer* window we have the *car* folder with a car model and the model we created called *VehicleCCU_Problem*. The *Model Explorer* window should be showing the top level folder of our *VehicleCCU_Problem* model that we just created (figure 7). If it does not then double-click on the *VehicleCCU_Problem* in the *Project Explorer* window OR select the *VehicleCCU_Problem* tab in the main window.


MyPapyrusProject - car/VehicleCCU_Problem.di - Papyrus
File Edit Navigate Search Papyrus Project Run Window Help
Quick Access
Project Explorer
car
car
VehicleCCU_Problem
car.di
VehicleCCU_Problem.di
Model Explorer
VehicleCCU_Problem
<Package Import> UML Primitive Types
<Package Import> Libraries
«EPackage, ModelLibrary» PrimitiveTypes
«ModelLibrary» Libraries
Aa
Outline
No outline for this editor
Notation Views
filter
View
Context
Create View
Welcome
Properties
Model Validation
Documentation
References
Properties are not available.
0 items selected

Internationalization

- [ ] Private storage
- [x] Use internationalization

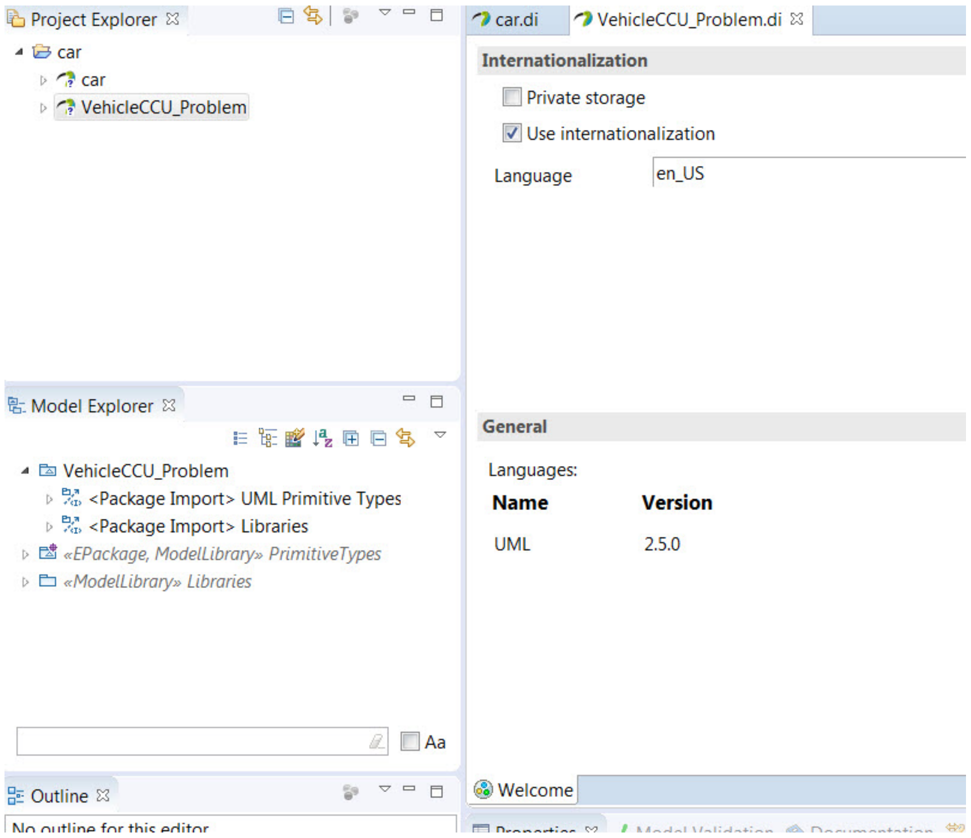
Language: en_US

General

Languages:

Name	Version
UML	2.5.0

Figure 6: Vehicle CCU Problem Model



Internationalization

- [ ] Private storage
- [x] Use internationalization

Language: en_US

General

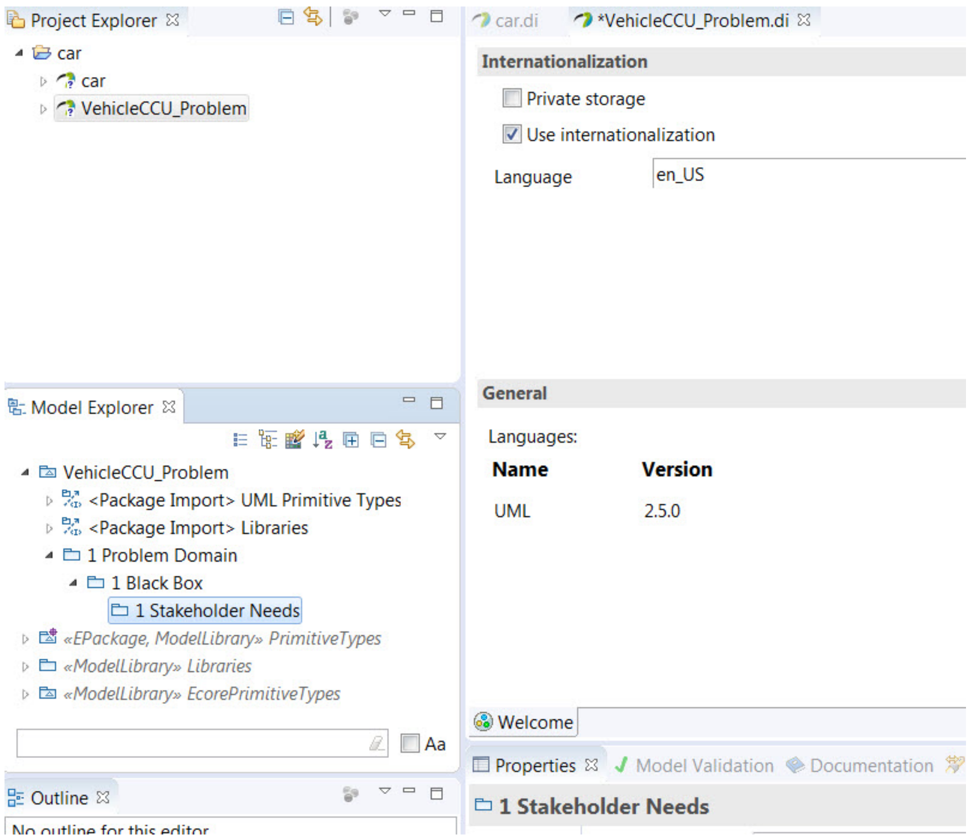
Languages:

Name	Version
UML	2.5.0

Figure 7: Vehicle CCU in Model Explorer

2.2.2 Step 2. Organize the model for Black-Box analysis.

1. We are now going to organize our model by creating three new packages in our *VehicleCCU_Problem* model and when we are finished it should look like figure 8.



Project Explorer
- car
  - car
  - VehicleCCU_Problem

car.di | *VehicleCCU_Problem.di

Internationalization

- [ ] Private storage
- [✓] Use internationalization

Language: en_US

Model Explorer
- VehicleCCU_Problem
  - <Package Import> UML Primitive Types
  - <Package Import> Libraries
  - 1 Problem Domain
    - 1 Black Box
      - 1 Stakeholder Needs
- «EPackage, ModelLibrary» PrimitiveTypes
- «ModelLibrary» Libraries
- «ModelLibrary» EcorePrimitiveTypes

General

Languages:

Name	Version
UML	2.5.0

Welcome

Properties | Model Validation | Documentation

1 Stakeholder Needs

Outline

No outline for this editor

Figure 8: Black Box Packages

2. Right-click on the *VehicleCCU_Problem* top level folder in the *Model Explorer* window and then select **New Child** and then **Package** (figure 9). The new package will appear under the *VehicleCCU_Problem* folder. Name the package *1 Problem Domain*. (Note that if you have already hit enter the package is named with a default name like *Package14* and it can be renamed in the UML properties tab at the bottom of the main window.

3. Now right-click on the *1 Problem Domain* package and follow the same procedure to create a new child package named *1 Black Box*.

4. Now right-click on the *1 Black Box* package and follow the same procedure to create a new child package named *1 Stakeholder Needs*. Your result should look like figure 8. If you make a mistake along the way you can simply delete a package and try again. Ensure that the packages are nested as shown in figure 8 before continuing.

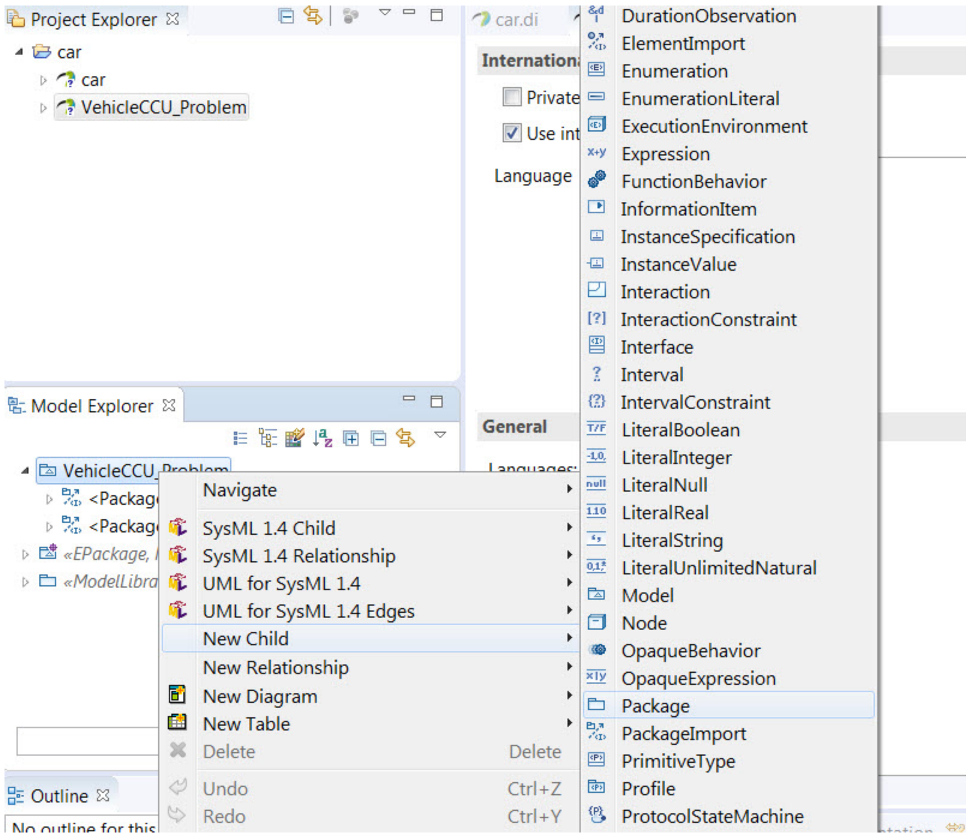
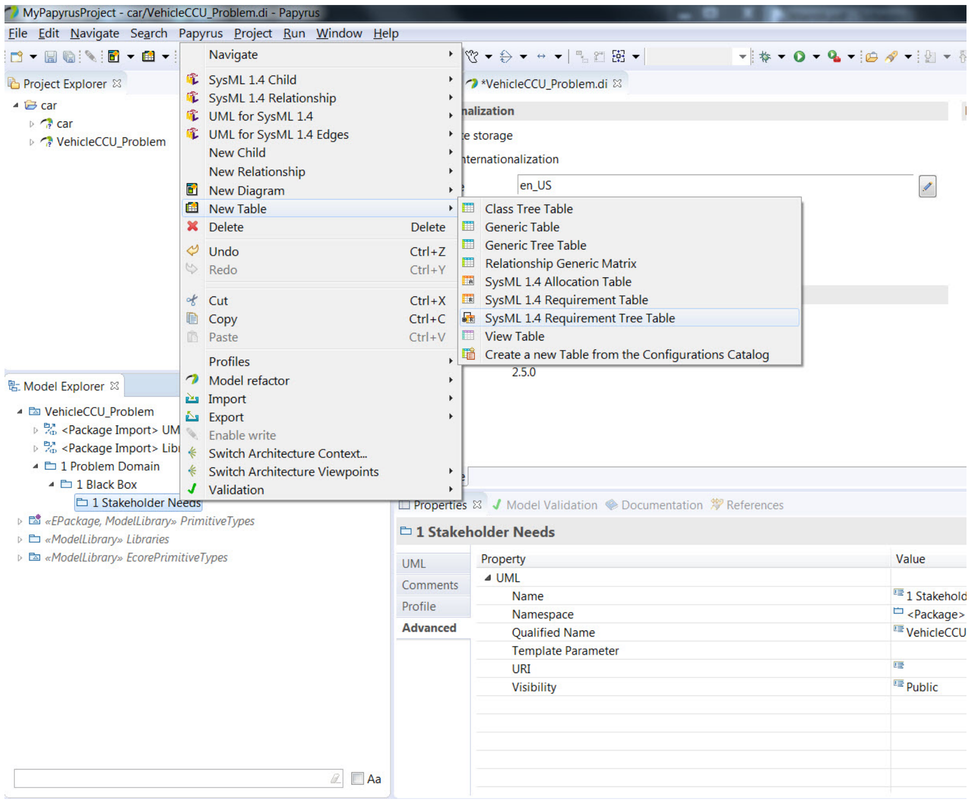


Figure 9: Create a New Package

2.2.3 Step 3. Capturing Stakeholder Needs (Requirements Table)

1. After talking with our stakeholders we will capture our initial list of stakeholder needs for the Vehicle Climate Control Unit using a SysML Requirement Table. There would of course be many requirements but we will just capture a few for illustrative purposes.

2. Right-click on the *1 Stakeholder Needs* package in the *Model Explorer* window and select **New Table > SysML 1.4 Requirement Tree Table** (figure 10). Name the table *Stakeholder Needs*. We want to use the Requirement Tree Table so we can have nested requirements in our requirements table.

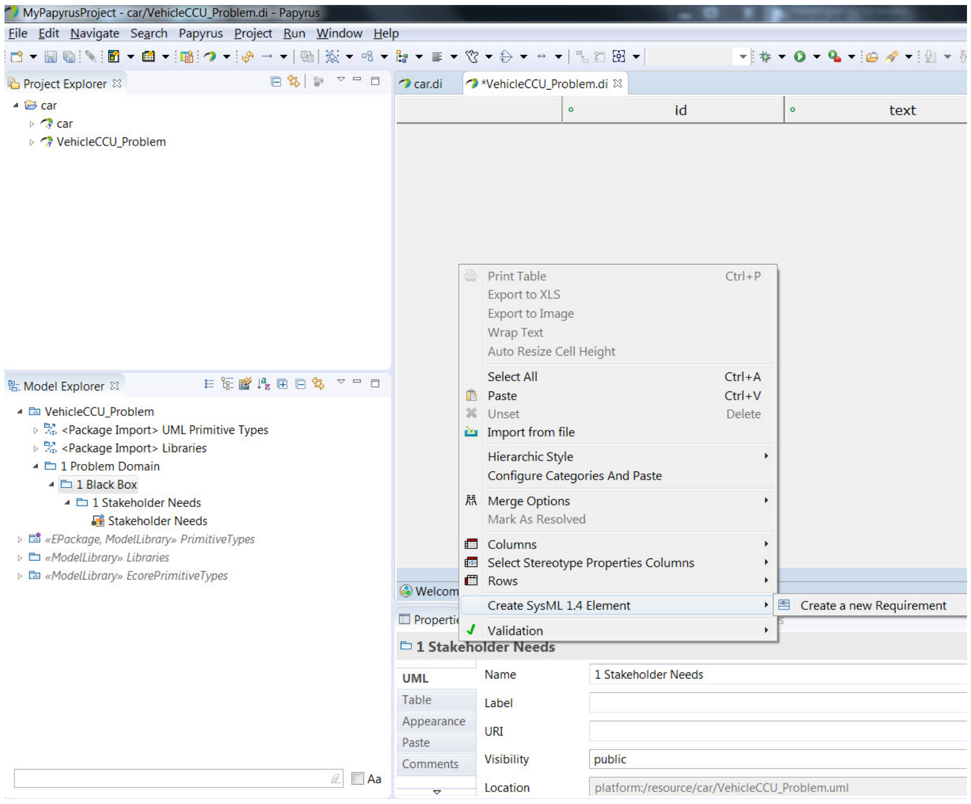


Property	Value
UML	
Name	1 Stakehold
Namespace	<Package>
Qualified Name	VehicleCCU
Template Parameter	
URI	
Visibility	Public

Figure 10: Creating a New Requirements Table

3. You should now see an initial table in the main window with two cells at the top labeled *id* and *text*.

4. First Left-click anywhere in the blank space in the main table window and then Right-click and select **Create SysML 1.4 Element > Create New Requirement** (see figure 11). A new requirement should appear in the table view and a requirement model element should appear underneath the *Stakeholder Needs* table element in the *Model Explorer* window (figure 12).

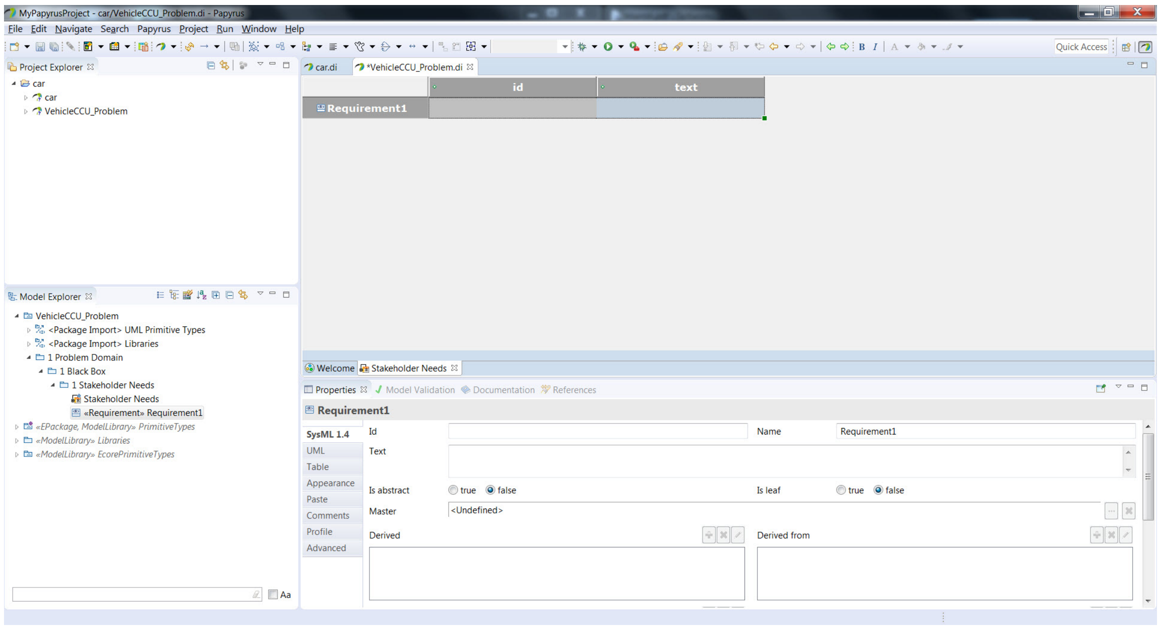


1 Stakeholder Needs

UML
Table
Appearance
Paste
Comments

Name	1 Stakeholder Needs
Label	
URI	
Visibility	public
Location	platform:/resource/car/VehicleCCU_Problem.uml

Figure 11: Create New Requirement in Requirement Table



Requirement1

SysML 1.4
UML
Table
Appearance
Paste
Comments
Profile
Advanced

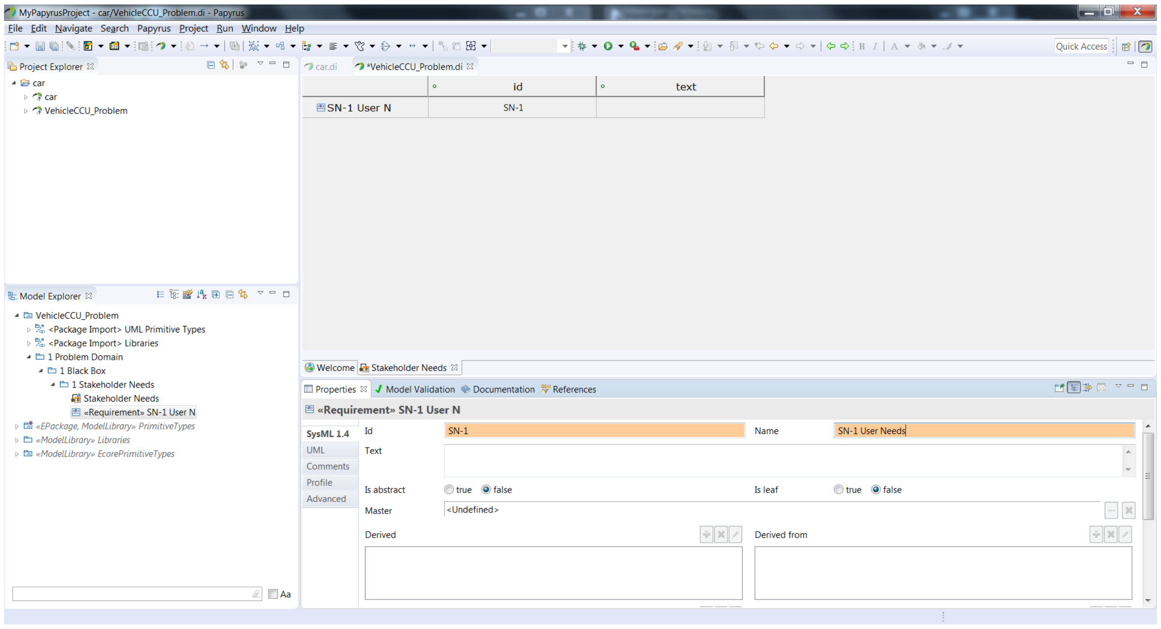
Id		Name	Requirement1
Text			
Is abstract	true [✓ false]	Is leaf	true [✓ false]
Master	<Undefined>		
Derived		Derived from	

Figure 12: New Requirement in Model

5. Most software like *Papyrus* has multiple ways to accomplish the same task. We can type directly in the requirements table cells to enter the *id* and *text* or we can enter it

in the *Properties Window* in the *SysML 1.4* tab. We are going to use the *SysML 1.4* tab at the bottom of the main screen . To do this Left-click on *Requirement 1* under the *Stakeholder Needs* table element in the *Model Explorer* then select the *SysML 1.4* tab in the *Properties* window (figure 12). You can see the properties in this tab for *Id*, *Name*, and *Text*.

6. Type *SN-1* in the *Id* field and *SN-1 User Needs* in the *Name* field. We will not enter anything in the *Text* field as this will be a heading for our user needs (figure 13).



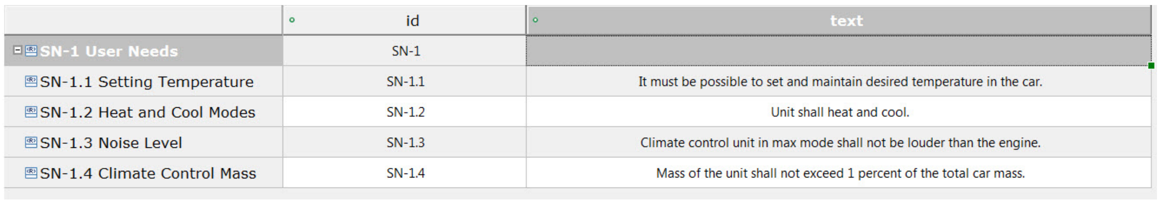
	id	text
SN-1 User N	SN-1	

«Requirement» SN-1 User N

Id: SN-1
Name: SN-1 User Needs
Text:
Is abstract: true [✓ false]
Is leaf: true [✓ false]
Master: <Undefined>
Derived
Derived from

Figure 13: User Needs

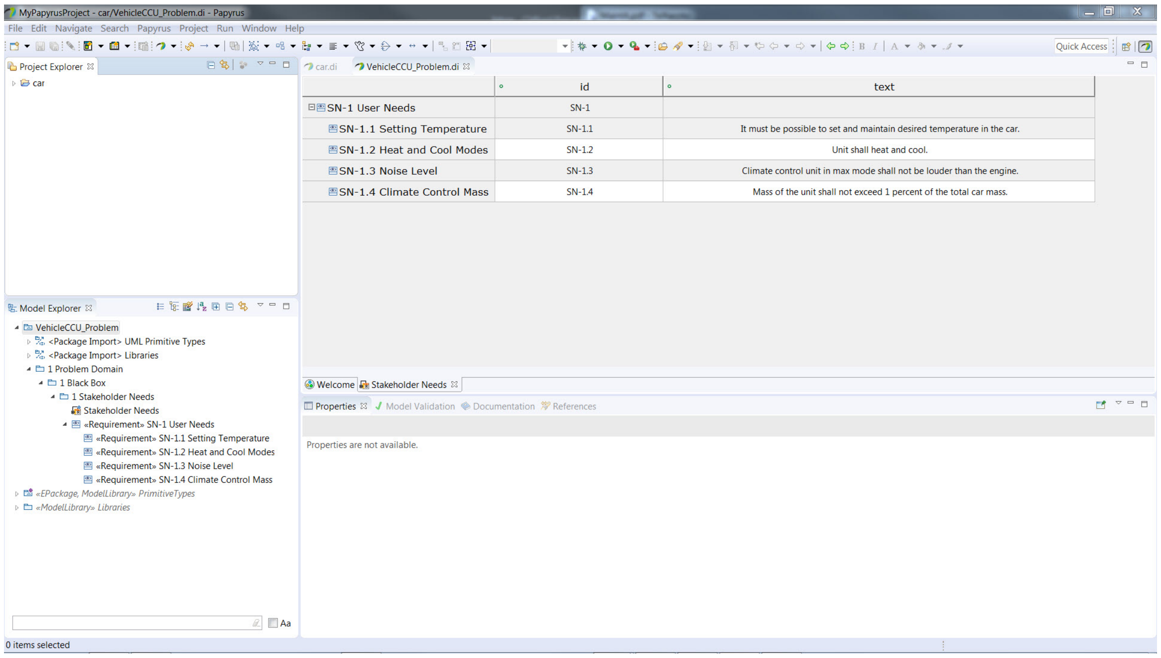
7. Continue adding requirements and populating the *Id*, *Name*, and *Text* fields until the requirements table looks like figure 14.



	id	text
SN-1 User Needs	SN-1	
SN-1.1 Setting Temperature	SN-1.1	It must be possible to set and maintain desired temperature in the car.
SN-1.2 Heat and Cool Modes	SN-1.2	Unit shall heat and cool.
SN-1.3 Noise Level	SN-1.3	Climate control unit in max mode shall not be louder than the engine.
SN-1.4 Climate Control Mass	SN-1.4	Mass of the unit shall not exceed 1 percent of the total car mass.

Figure 14: User Needs Requirements Table

8. In the *Model View* window Left-click on the requirement *SN-1.1 Setting Temperature* requirement then hold down the Control key and select the last requirement *SN-1.4 Climate Control Mass*. Drag these requirements and drop them on the *SN-1 User Needs* to make these requirements child requirements of our heading requirement. Your screen should look like figure 15 with the requirements nested under *SN-1 User Needs* in the *Model View* window and a tree structure in the table view.



	id	text
SN-1 User Needs	SN-1	
SN-1.1 Setting Temperature	SN-1.1	It must be possible to set and maintain desired temperature in the car.
SN-1.2 Heat and Cool Modes	SN-1.2	Unit shall heat and cool.
SN-1.3 Noise Level	SN-1.3	Climate control unit in max mode shall not be louder than the engine.
SN-1.4 Climate Control Mass	SN-1.4	Mass of the unit shall not exceed 1 percent of the total car mass.

```
VehicleCCU_Problem
  <Package Import> UML Primitive Types
  <Package Import> Libraries
  1 Problem Domain
    1 Black Box
      1 Stakeholder Needs
        Stakeholder Needs
        «Requirement» SN-1 User Needs
          «Requirement» SN-1.1 Setting Temperature
          «Requirement» SN-1.2 Heat and Cool Modes
          «Requirement» SN-1.3 Noise Level
          «Requirement» SN-1.4 Climate Control Mass
«EPackage, ModelLibrary» PrimitiveTypes
«ModelLibrary» Libraries
```

Properties are not available.

0 items selected

Figure 15: Requirements Tree

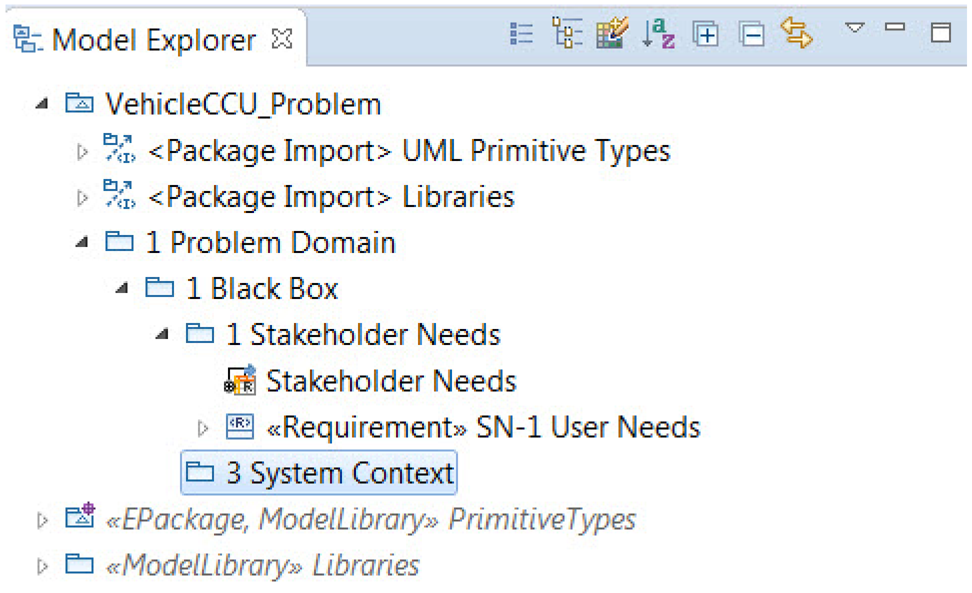
2.2.4 Step 4. Initial System Context

The operating environment or System Context defines the external view of the system where the SoI is treated as a black box. It introduces elements that interact with the system but are not owned by the system. During this initial phase of system context definition one or more system contexts are defined and elements that participate in each context are identified. Several sources recommend using blocks rather than actors to define humans for the following reasons:

- A block is treated as a part of the system context. An actor is not.
- A block can be marked as external. An actor cannot.
- A block can have its behavior defined. An actor cannot.

Let's start modeling the Initial System Context.

1. Set up a new package under the *1 Black Box* package named *3 System Context* by Right-clicking on the *1 Black Box* package in the *Model Explorer* window and selecting **New Child** and then **Package**. Name the new package *3 System Context* (figure 16).

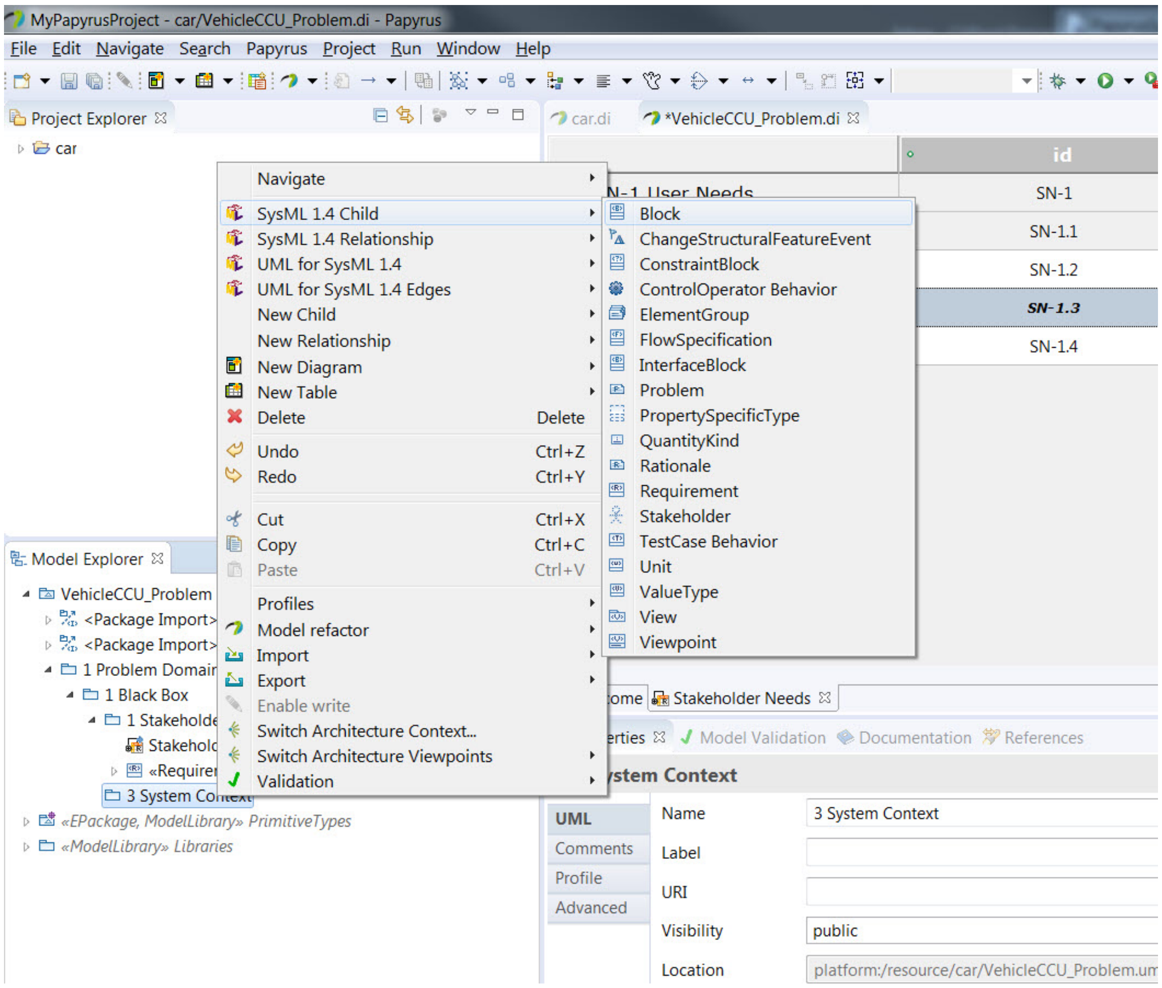


Model Explorer

- VehicleCCU_Problem
  - <Package Import> UML Primitive Types
  - <Package Import> Libraries
  - 1 Problem Domain
    - 1 Black Box
      - 1 Stakeholder Needs
        - Stakeholder Needs
        - «Requirement» SN-1 User Needs
    - 3 System Context
- *«EPackage, ModelLibrary» PrimitiveTypes*
- *«ModelLibrary» Libraries*

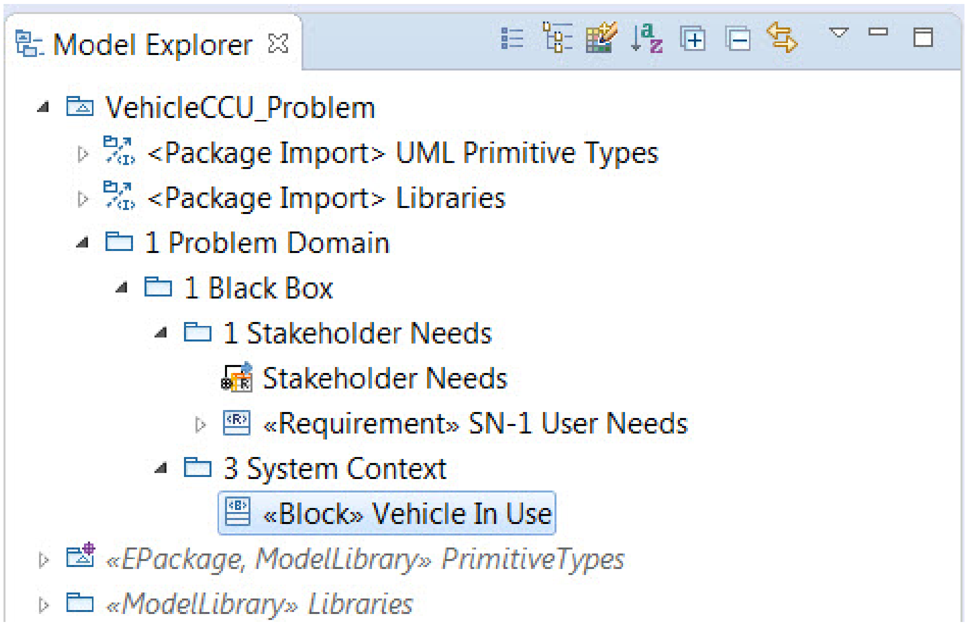
Figure 16: System Context package

2. Now we will create a block that captures the vehicle in use system context. Right-click on the *3 System Context* package just created and select **SysML 1.4 Child** and then **Block** (figure 17). Name the block *Vehicle In Use*. Your model explorer should now look like figure 18.



Name	3 System Context
Label	
URI	
Visibility	public
Location	platform:/resource/car/VehicleCCU_Problem.ur

Figure 17: Create New Block


Model Explorer

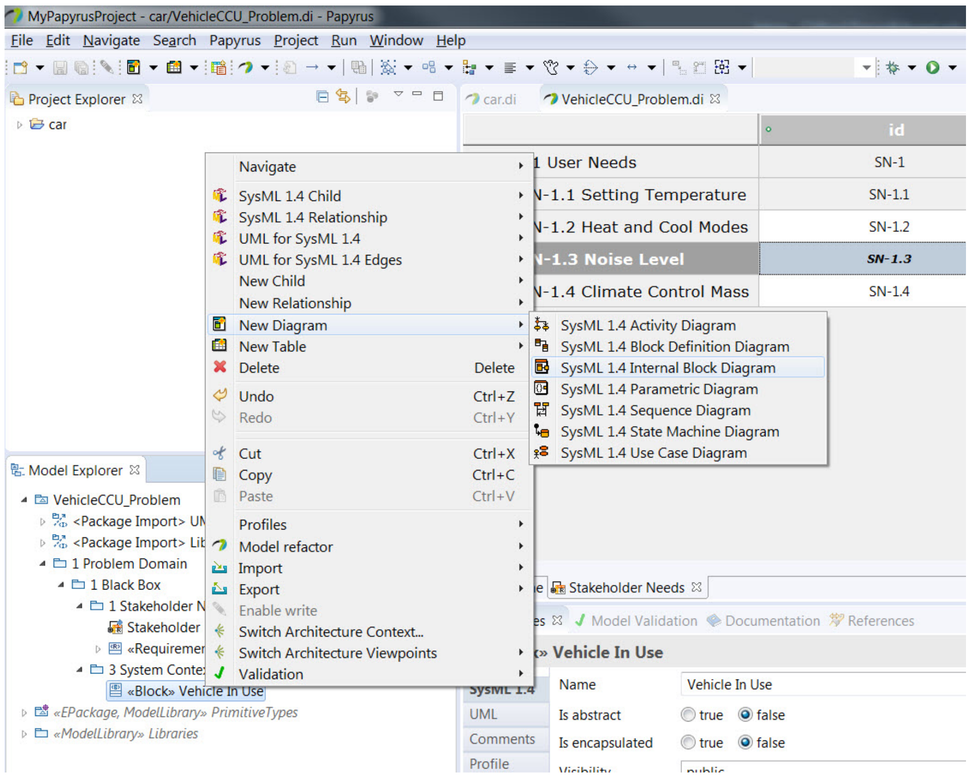
VehicleCCU_Problem
- <Package Import> UML Primitive Types
- <Package Import> Libraries
- 1 Problem Domain
  - 1 Black Box
    - 1 Stakeholder Needs
      - Stakeholder Needs
      - «Requirement» SN-1 User Needs
    - 3 System Context
      - «Block» Vehicle In Use

«EPackage, ModelLibrary» PrimitiveTypes

«ModelLibrary» Libraries

Figure 18: Vehicle In Use Block

3. To specify participants of the Vehicle In Use context we will create a SysML Internal Block Diagram (IBD). Right-click on the *Vehicle In Use* block just created and select **New Diagram** then **SysML 1.4 Internal Block Diagram** (figure 19). Name the new diagram *Vehicle In Use*, the same as the block. You should see the diagram in the main window (figure 20). If you do not, simply Double-click on the *Vehicle In Use* internal block diagram in the *Model Explorer* window.

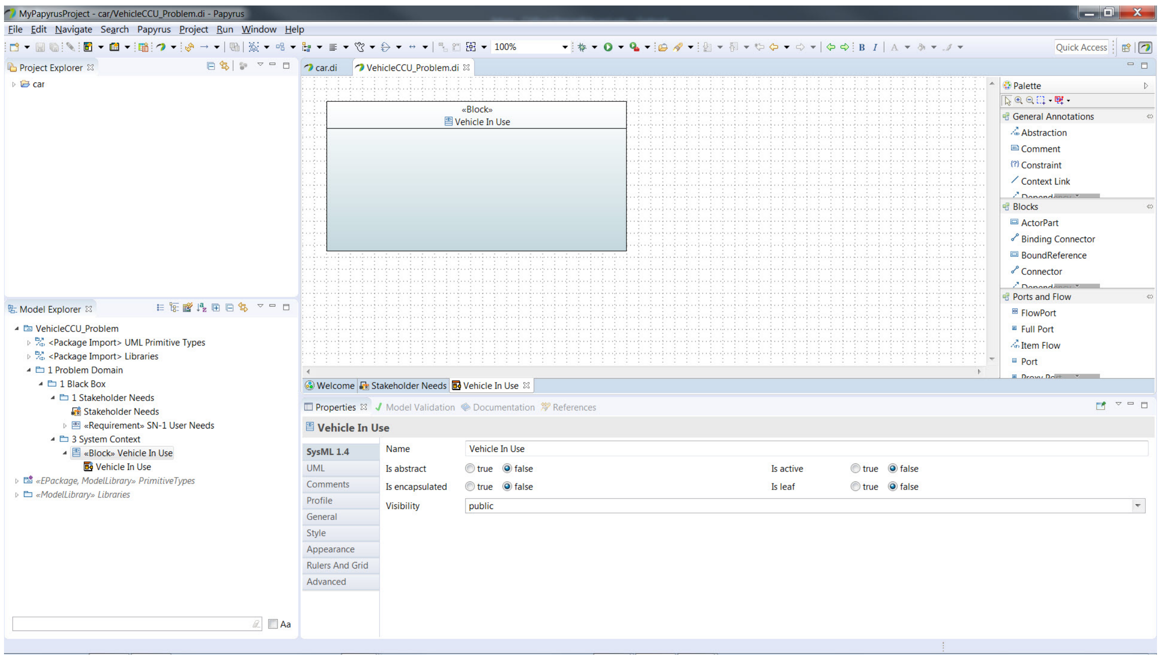


	id
1 User Needs	SN-1
N-1.1 Setting Temperature	SN-1.1
N-1.2 Heat and Cool Modes	SN-1.2
N-1.3 Noise Level	SN-1.3
N-1.4 Climate Control Mass	SN-1.4

Vehicle In Use

Name	Vehicle In Use
Is abstract	true [✓ false]
Is encapsulated	true [✓ false]

Figure 19: Create Internal Block Diagram



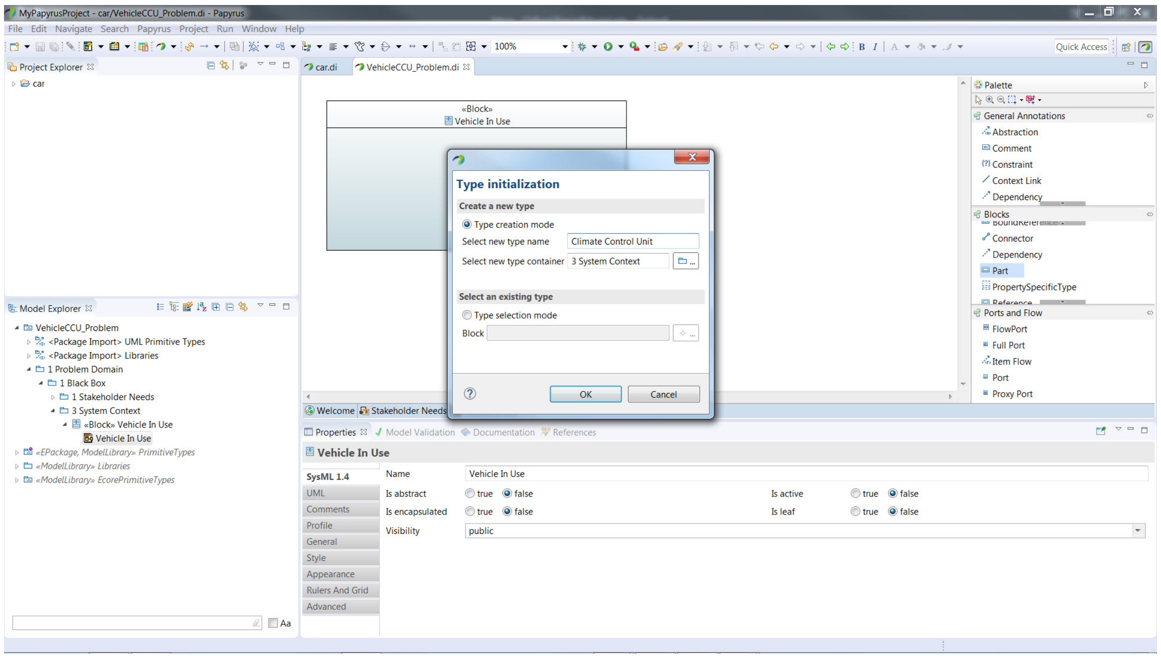
Vehicle In Use

Name	Vehicle In Use		
Is abstract	true [✓ false]	Is active	true [✓ false]
Is encapsulated	true [✓ false]	Is leaf	true [✓ false]
Visibility	public		

Figure 20: Vehicle In Use Internal Block Diagram

4. Our analysis of stakeholder needs indicates that the Vehicle In Use system context includes the following participants: the vehicle Climate Control Unit, Vehicle Occu-

pant, and an Energy Supply of some kind to power everything. The participants can be captured as part properties in the IBD. Open the *Vehicle In Use* IBD you just created if not already open. Use the tool palette (If the tool palette is not visible click on the small arrow on the right side of the main window vertical scroll bar to display it) on the right and select **Part** from the **Blocks** diagram palette . Now Left-click anywhere inside the *Vehicle In Use* diagram in the main window and the *Type Initialization* window will appear (figure 21).



Type initialization

Create a new type

(●) Type creation mode

Select new type name: Climate Control Unit

Select new type container: 3 System Context

Select an existing type

( ) Type selection mode

Block:

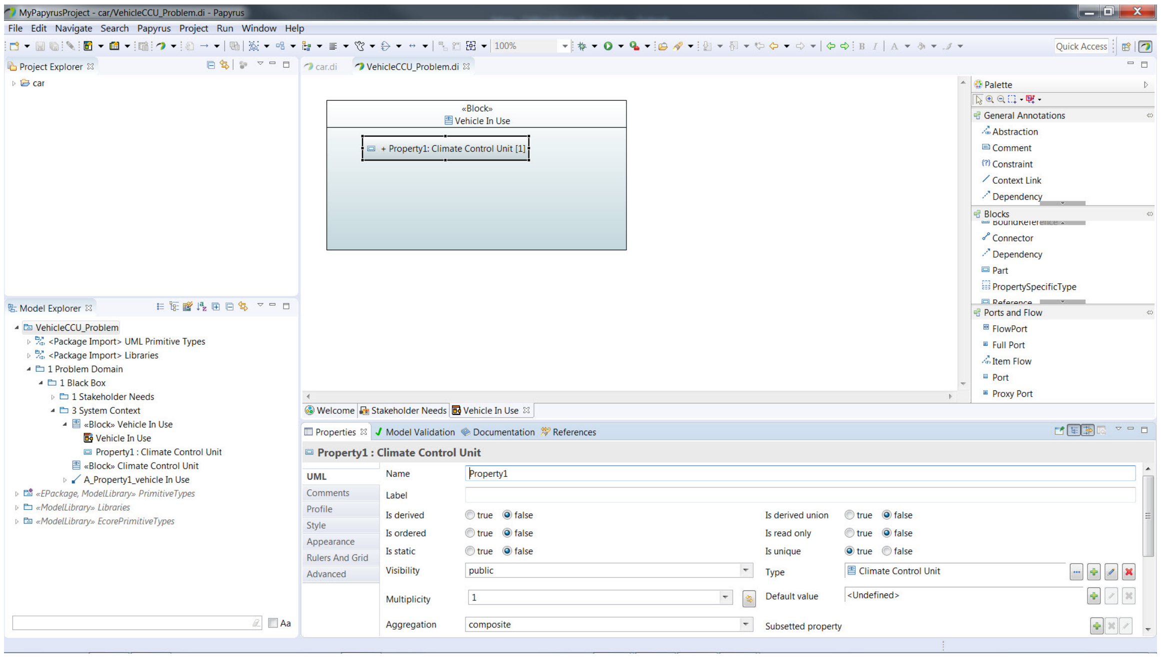
OK Cancel

Vehicle In Use

Name	Vehicle In Use		
Is abstract	( ) true (●) false	Is active	( ) true (●) false
Is encapsulated	( ) true (●) false	Is leaf	( ) true (●) false
Visibility	public		

Figure 21: Part Type Initialization Window

5. In the *Type Initialization* window ensure that the *Type creation mode* radio button is selected to create a new type. In the *Select new type name box* type *Climate Control Unit* and press *Enter* twice. Your model should look like figure 22.

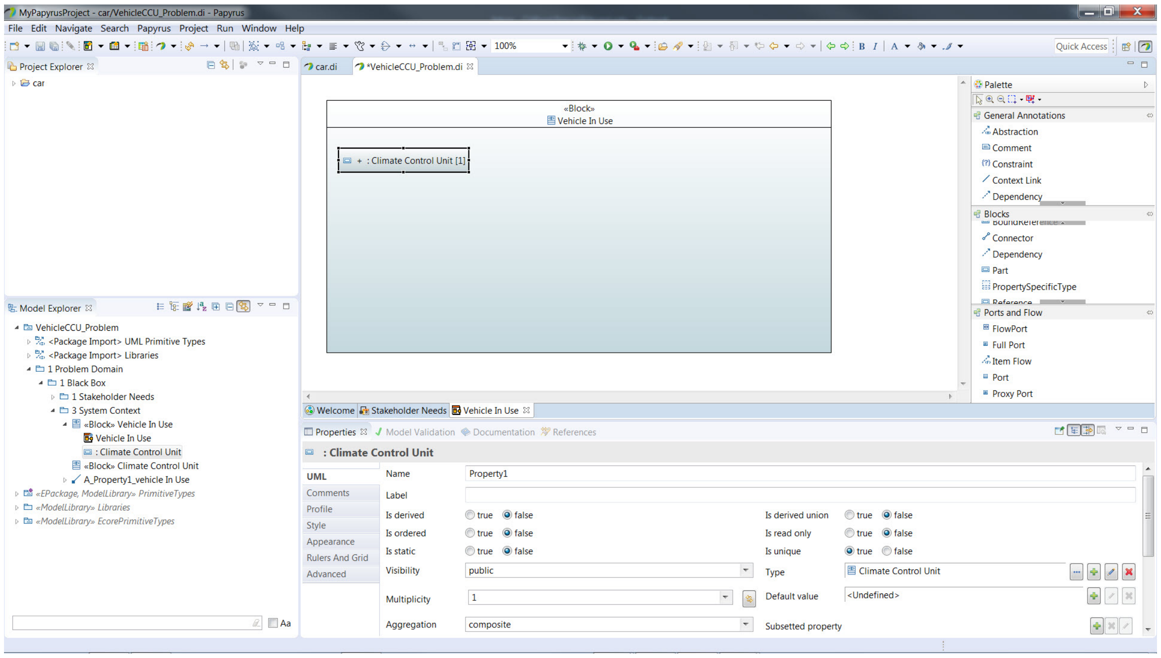


Property1 : Climate Control Unit

Name	Property1		
Label			
Is derived	true [✓ false]	Is derived union	true [✓ false]
Is ordered	true [✓ false]	Is read only	true [✓ false]
Is static	true [✓ false]	Is unique	[✓ true] false
Visibility	public	Type	Climate Control Unit
Multiplicity	1	Default value	<Undefined>
Aggregation	composite	Subsetted property	

Figure 22: Part Type Initialization Window

6. Let's make this new block a little cleaner by not displaying the *Property1* name in front of the colon in the *Climate Control Unit* part in the diagram. Select the *Climate Control Unit* in the diagram. In the *Properties* window at the bottom of the screen select the *UML* tab and place your cursor in the *Label* field. Type a single *Space* with the spacebar and hit *Enter*. Your *Climate Control Unit* should now look like figure 23.

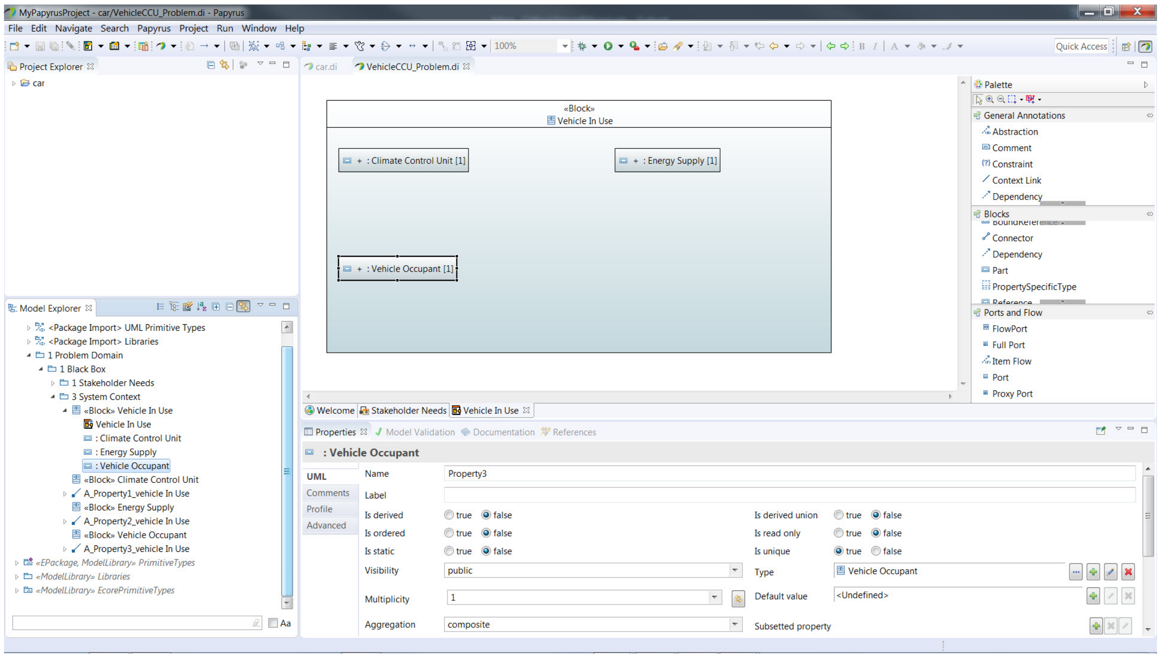


: Climate Control Unit

Name	Property1		
Label			
Is derived	true [✓ false]	Is derived union	true [✓ false]
Is ordered	true [✓ false]	Is read only	true [✓ false]
Is static	true [✓ false]	Is unique	[✓ true] false
Visibility	public	Type	Climate Control Unit
Multiplicity	1	Default value	<Undefined>
Aggregation	composite	Subsetted property	

Figure 23: Climate Control Unit with Label Removed

7. Create two more parts; one called *Vehicle Occupant* and one called *Energy Supply*. When finished your diagram should look like figure 24.



Model Explorer

- <Package Import> UML Primitive Types
- <Package Import> Libraries
- 1 Problem Domain
  - 1 Black Box
    - 1 Stakeholder Needs
    - 3 System Context
      - «Block» Vehicle In Use
        - Vehicle In Use
        - : Climate Control Unit
        - : Energy Supply
        - : Vehicle Occupant
      - «Block» Climate Control Unit
      - A_Property1_vehicle In Use
      - «Block» Energy Supply
      - A_Property2_vehicle In Use
      - «Block» Vehicle Occupant
      - A_Property3_vehicle In Use
- *«EPackage, ModelLibrary» PrimitiveTypes*
- *«ModelLibrary» Libraries*
- *«ModelLibrary» EcorePrimitiveTypes*

Properties | Model Validation | Documentation | References

: Vehicle Occupant

UML | Comments | Profile | Advanced

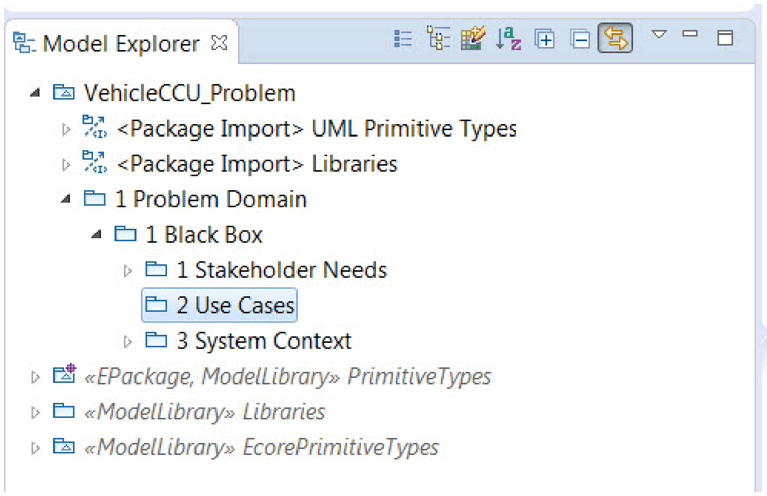
Name	Property3		
Label			
Is derived	true [✓ false]	Is derived union	true [✓ false]
Is ordered	true [✓ false]	Is read only	true [✓ false]
Is static	true [✓ false]	Is unique	[✓ true] false
Visibility	public	Type	Vehicle Occupant
Multiplicity	1	Default value	<Undefined>
Aggregation	composite	Subsetted property	

Figure 24: Vehicle In Use IBD

2.2.5 Step 5. Use Case and Activity Diagram

In this section, the functional stakeholder needs will be refined with use cases and use case scenarios. Again we will be using blocks rather than actors (see section 2.2.4).

1. Set up a new package under the *1 Black Box* package named *2 Use Cases* by Right-clicking on the *1 Black Box* package in the *Model Explorer* window and selecting **New Child** and then **Package**. Name the new package *2 Use Cases* (figure 25).

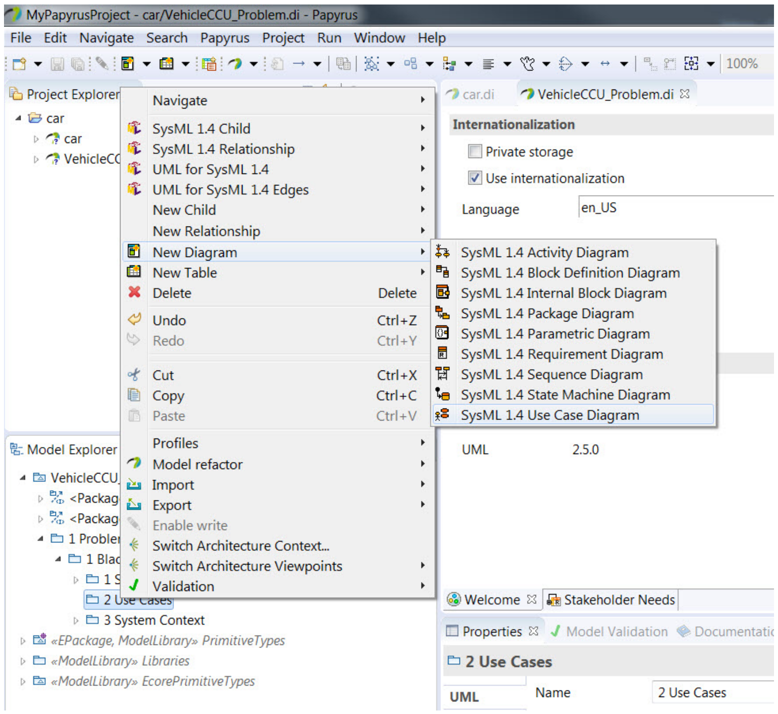


Model Explorer

- VehicleCCU_Problem
  - <Package Import> UML Primitive Types
  - <Package Import> Libraries
  - 1 Problem Domain
    - 1 Black Box
      - 1 Stakeholder Needs
      - 2 Use Cases
      - 3 System Context
- *«EPackage, ModelLibrary» PrimitiveTypes*
- *«ModelLibrary» Libraries*
- *«ModelLibrary» EcorePrimitiveTypes*

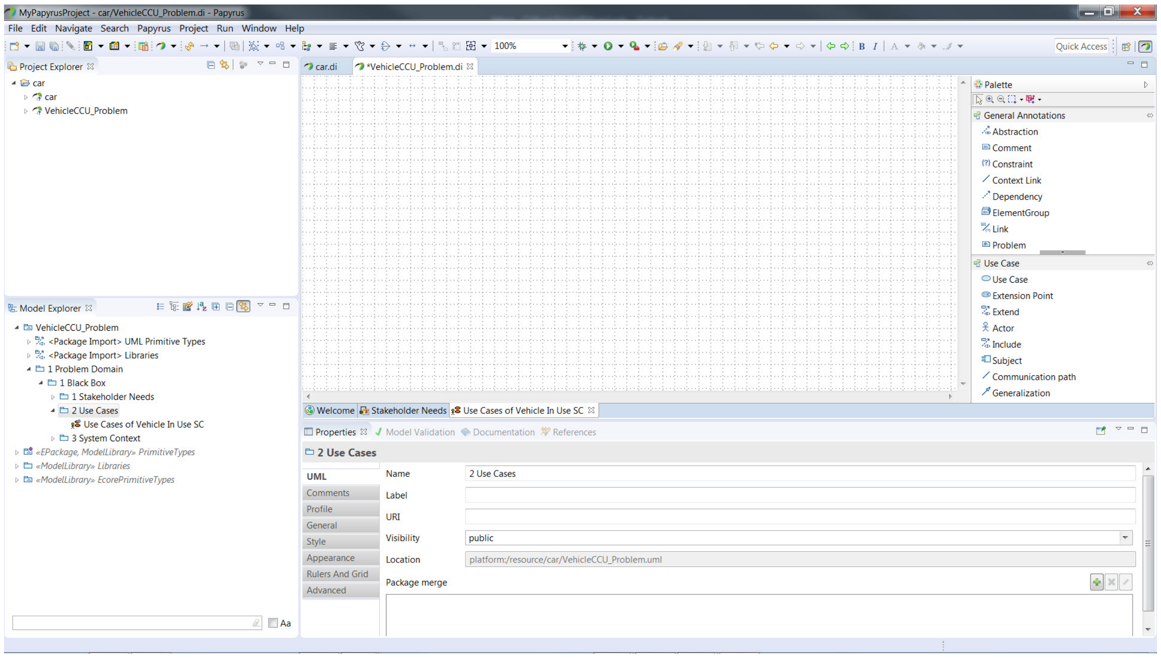
Figure 25: Use Case Package

2. Now that we have created the use case package we need to create a Use Case Diagram. In the *Model Explorer* window, Right-click on the *2 Use Cases* package that you created in the previous step and select *New Diagram* and then *SysML 1.4 Use Case Diagram* (figure 26). Name the diagram *Use Cases of Vehicle In Use SC*. The new use case diagram will appear under your *2 Use Cases* package and a new blank use case diagram should appear in the main window (figure 27).



[ ] Private storage
[✓] Use internationalization
Language en_US
UML 2.5.0
Name 2 Use Cases

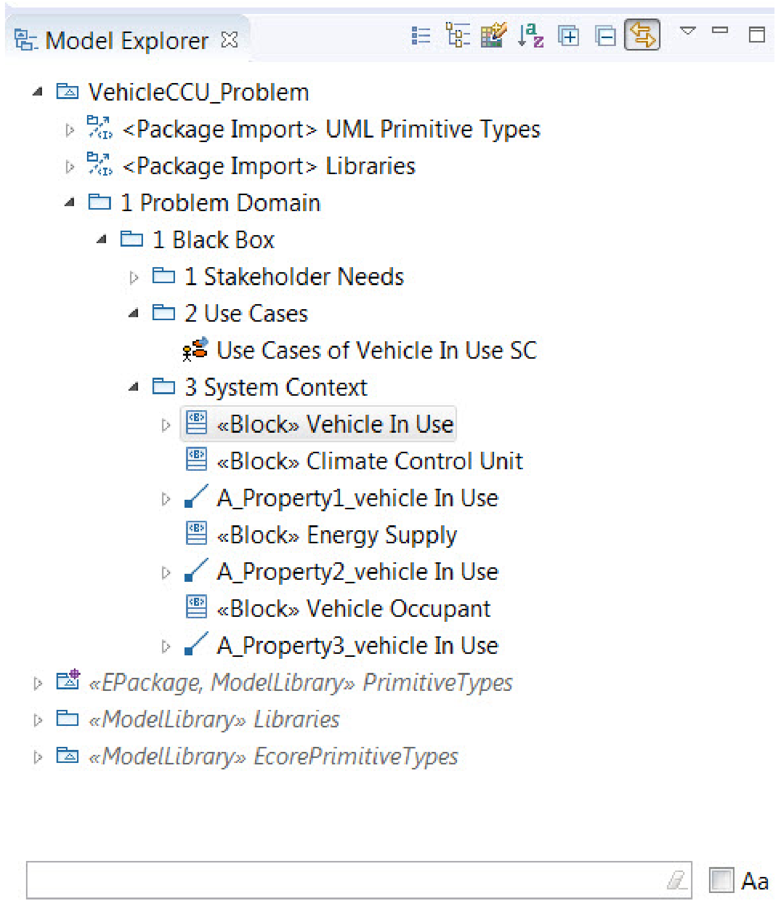
Figure 26: Use Case Diagram Creation



Name	2 Use Cases
Label	
URI	
Visibility	public
Location	platform:/resource/car/VehicleCCU_Problem.uml

Figure 27: New Use Case Diagram

3. In the *Model Explorer* window, click on the arrow to the left of the *3 System Context* package to expand its contents (figure 28).



Model Explorer

VehicleCCU_Problem
- <Package Import> UML Primitive Types
- <Package Import> Libraries
- 1 Problem Domain
  - 1 Black Box
    - 1 Stakeholder Needs
    - 2 Use Cases
      - Use Cases of Vehicle In Use SC
    - 3 System Context
      - «Block» Vehicle In Use
      - «Block» Climate Control Unit
      - A_Property1_vehicle In Use
      - «Block» Energy Supply
      - A_Property2_vehicle In Use
      - «Block» Vehicle Occupant
      - A_Property3_vehicle In Use

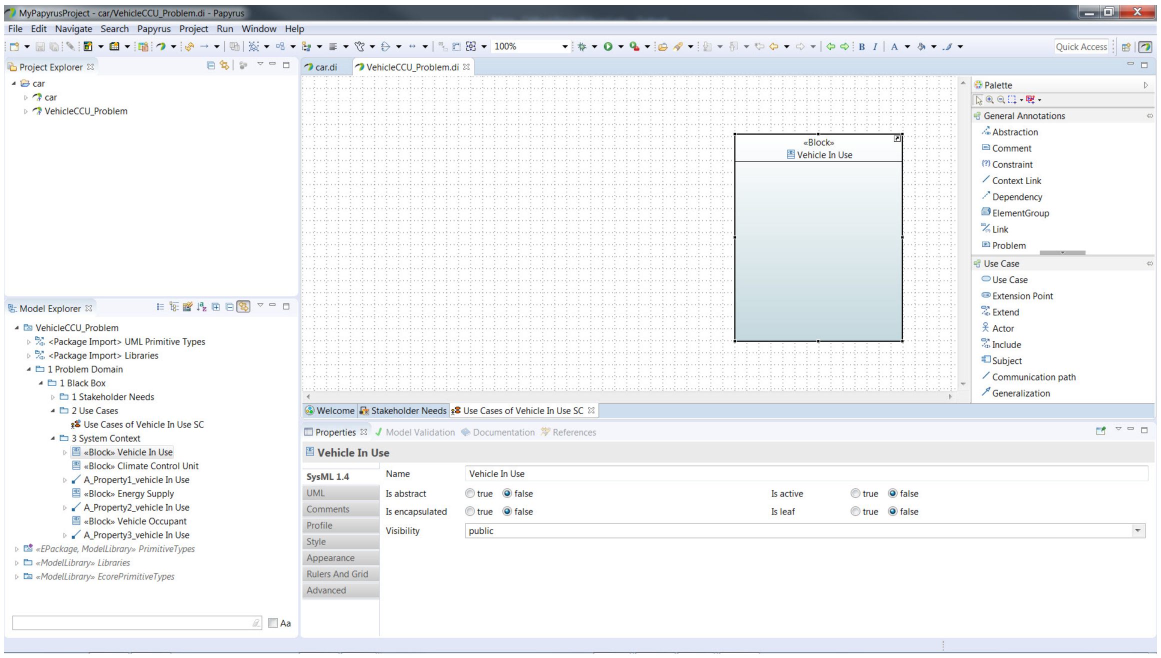
«EPackage, ModelLibrary» PrimitiveTypes

«ModelLibrary» Libraries

«ModelLibrary» EcorePrimitiveTypes

Figure 28: Expand the System Context package

4. Click and Drag the *Vehicle In Use* block onto the *Use Cases of Vehicle In Use* use case diagram in the main window. A copy of the block should appear in the use case diagram (figure 29).


Vehicle In Use

SysML 1.4

Name: Vehicle In Use

Is abstract: true [✓ false]

Is active: true [✓ false]

Is encapsulated: true [✓ false]

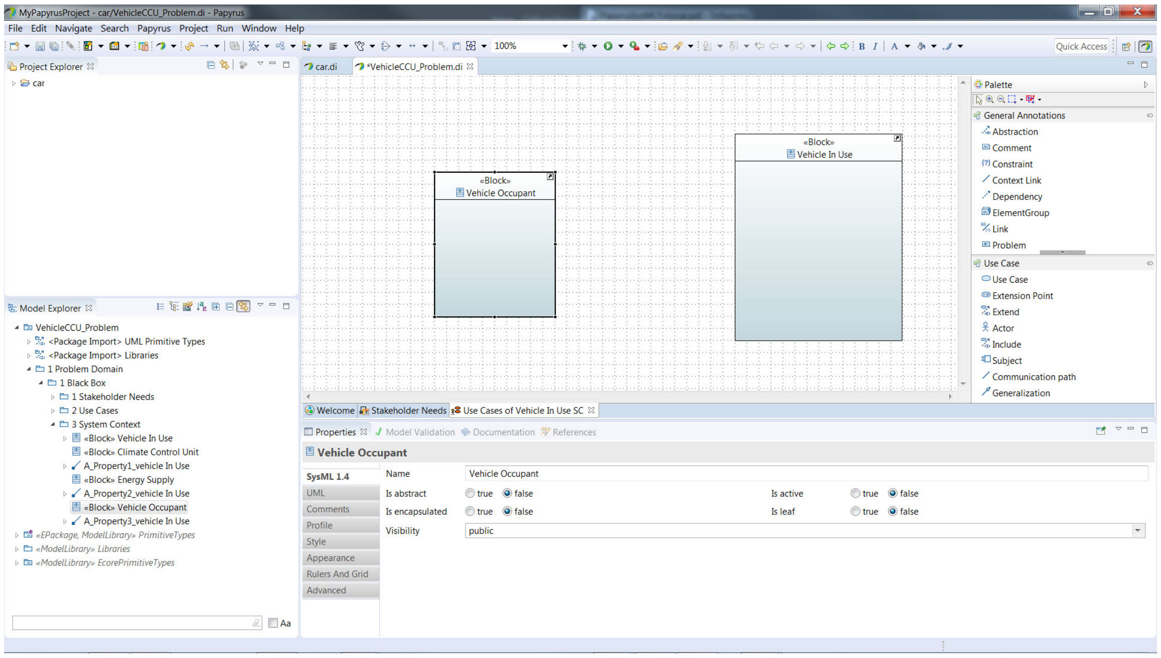
Is leaf: true [✓ false]

Visibility: public

MyPapyrusProject - car/VehicleCCU_Problem.di - Papyrus
File Edit Navigate Search Papyrus Project Run Window Help
Project Explorer
Model Explorer
VehicleCCU_Problem
<Package Import> UML Primitive Types
<Package Import> Libraries
1 Problem Domain
1 Black Box
1 Stakeholder Needs
2 Use Cases
Use Cases of Vehicle In Use SC
3 System Context
«Block» Vehicle In Use
«Block» Climate Control Unit
A_Property1_vehicle In Use
«Block» Energy Supply
A_Property2_vehicle In Use
«Block» Vehicle Occupant
A_Property3_vehicle In Use
«Block»
Vehicle In Use
Palette
General Annotations
Use Case
Welcome
Stakeholder Needs
Use Cases of Vehicle In Use SC
Properties

Figure 29: Vehicle In Use block in Use Case Diagram

5. Click and Drag the *Vehicle Occupant* block onto the *Use Cases of Vehicle In Use* use case diagram in the main window. A copy of the block should appear in the use case diagram (figure 30).


Vehicle Occupant

SysML 1.4

Name: Vehicle Occupant

Is abstract: true [✓ false]

Is active: true [✓ false]

Is encapsulated: true [✓ false]

Is leaf: true [✓ false]

Visibility: public

MyPapyrusProject - car/VehicleCCU_Problem.di - Papyrus
*VehicleCCU_Problem.di
«Block»
Vehicle Occupant
«Block»
Vehicle In Use
Use Cases of Vehicle In Use SC

Figure 30: Vehicle Occupant block in Use Case Diagram

6. In our analysis of stakeholder needs we have identified the *Feel Comfortable Temperature* use case performed in the *Vehicle In Use* system context. We will now capture this use case in the *Use Cases of Vehicle In Use SC* diagram. The tool palette on the right should be displaying the Use Case tool palette. Left-click on the *Use Case* item (figure 31) and then Left-click anywhere in the *Vehicle In Use* block. A use case is created in this block. Name this use case *Feel Comfortable Temperature* and press Enter. Your screen should look like (figure 32).

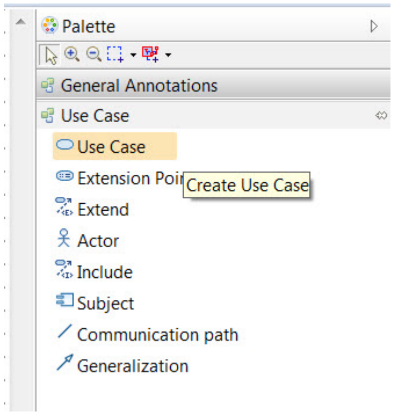
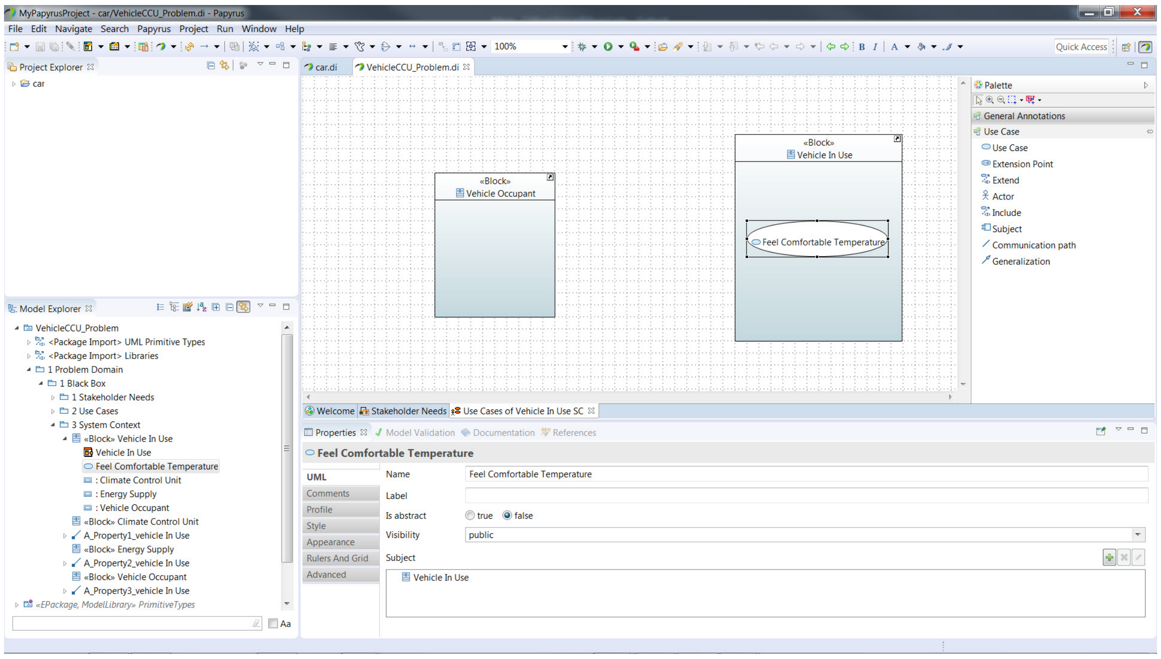


Figure 31: Use Case Tool Palette



Feel Comfortable Temperature

Name	Feel Comfortable Temperature
Label	
Is abstract	true [✓ false]
Visibility	public
Subject	Vehicle In Use

Figure 32: Vehicle Occupant block in Use Case Diagram

7. Now we will create an association between the *Feel Comfortable Temperature* use case and the *Vehicle Occupant* block (recall that we are using a block here to represent the traditional UML actor for the reasons stated earlier in section 2.2.4). Left-

click the *Feel Comfortable Temperature* use case in the diagram. Now Left-click the *Communication Path* item in the *Use Case* tool palette (figure 33). Now Left-click on the *Feel Comfortable Temperature* use case and then Left-click on the *Vehicle Occupant* block in the diagram. An association should be created and your model should look like (figure 34).

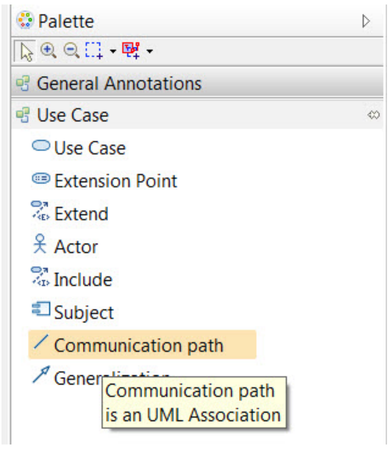
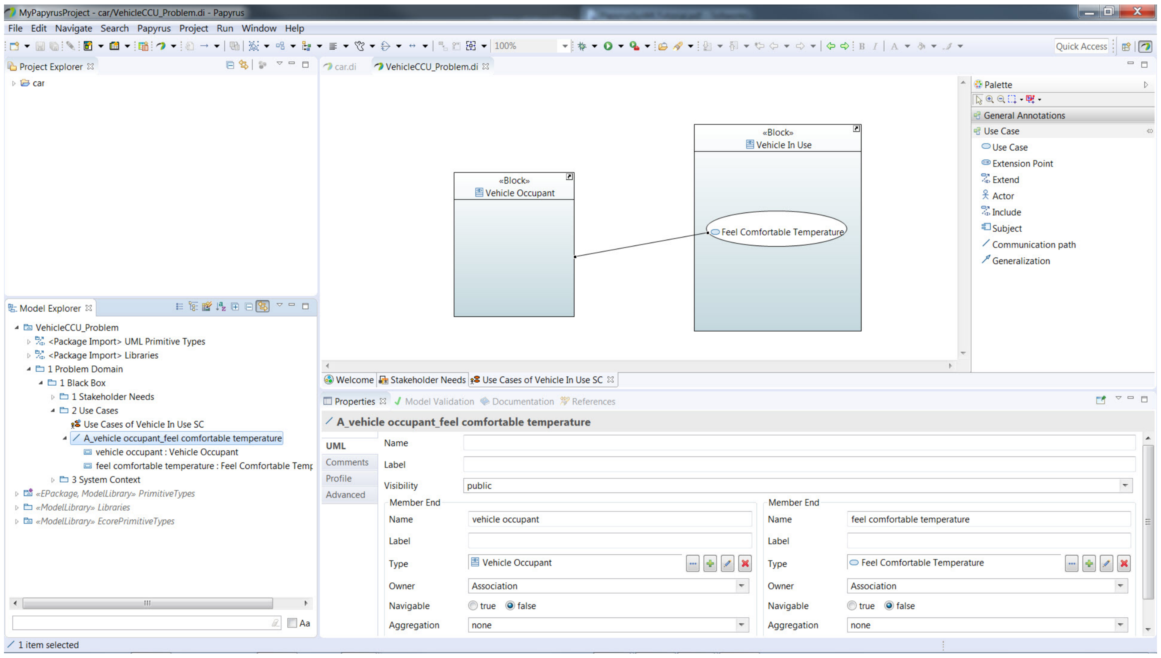


Figure 33: Use Case Tool Palette


MyPapyrusProject - car/VehicleCCU_Problem.di - Papyrus
File Edit Navigate Search Papyrus Project Run Window Help
Project Explorer
car
car.di
VehicleCCU_Problem.di
«Block»
Vehicle In Use
«Block»
Vehicle Occupant
Feel Comfortable Temperature
Palette
General Annotations
Use Case
Use Case
Extension Point
Extend
Actor
Include
Subject
Communication path
Generalization
Model Explorer
VehicleCCU_Problem
<Package Import> UML Primitive Types
<Package Import> Libraries
1 Problem Domain
1 Black Box
1 Stakeholder Needs
2 Use Cases
Use Cases of Vehicle In Use SC
A_vehicle occupant_feel comfortable temperature
vehicle occupant : Vehicle Occupant
feel comfortable temperature : Feel Comfortable Temp
3 System Context
«EPackage, ModelLibrary» PrimitiveTypes
«ModelLibrary» Libraries
«ModelLibrary» EcorePrimitiveTypes
Welcome
Stakeholder Needs
Use Cases of Vehicle In Use SC
Properties
Model Validation
Documentation
References
UML
Comments
Profile
Advanced
1 item selected

A_vehicle occupant_feel comfortable temperature

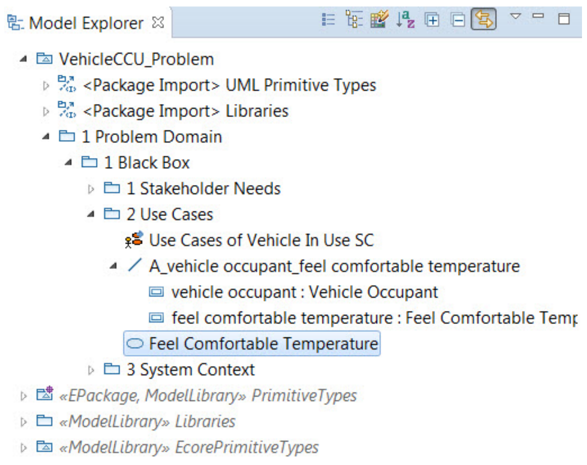
Name:
Label:
Visibility: public

Member End		Member End	
Name	vehicle occupant	Name	feel comfortable temperature
Label		Label	
Type	Vehicle Occupant	Type	Feel Comfortable Temperature
Owner	Association	Owner	Association
Navigable	true [✓ false]	Navigable	true [✓ false]
Aggregation	none	Aggregation	none

Figure 34: Association Created

8. Now we will elaborate the *Feel Comfortable Temperature* use case as an activity diagram. First let's move the *Feel Comfortable Temperature* use case from the *3 System Context* package and place it under the *2 Use Cases* package. To do this, expand the *3 System Context* package view in the *Model Explorer* window. Now

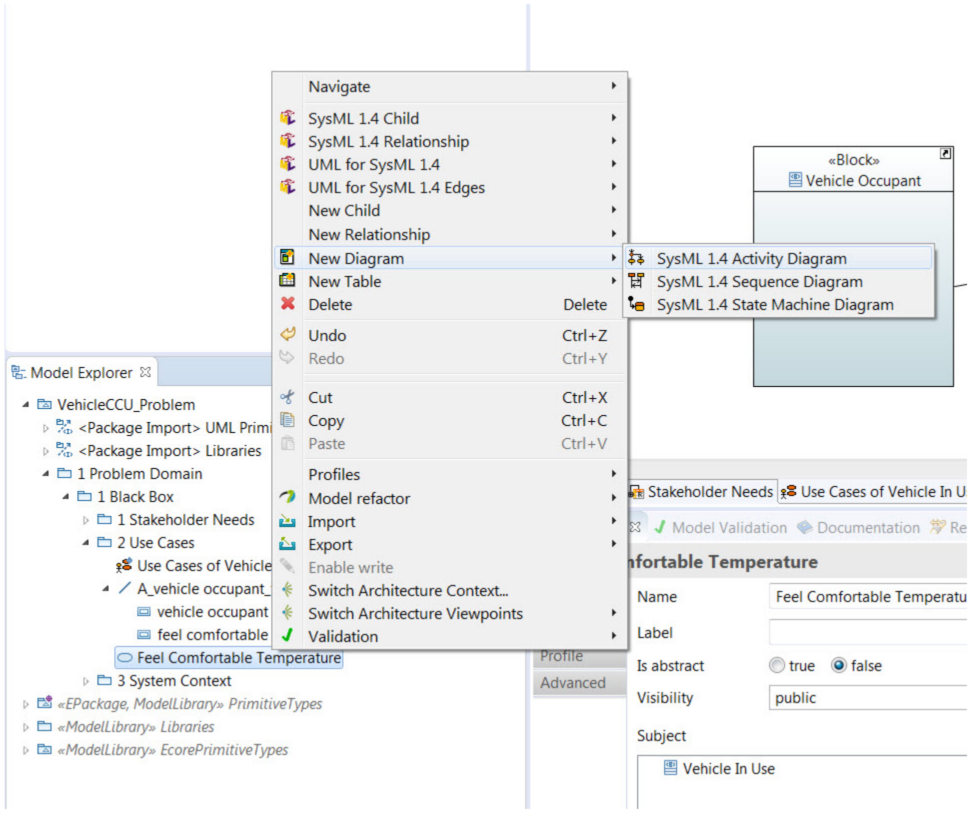
Click and Drag the *Feel Comfortable Temperature* use case and Drop it on the *2 Use Cases* package in the *Model Explorer* window. The use case is now a child of the *2 Use Cases* package (figure 35).


Model Explorer

```
VehicleCCU_Problem
  <Package Import> UML Primitive Types
  <Package Import> Libraries
  1 Problem Domain
    1 Black Box
      1 Stakeholder Needs
      2 Use Cases
        Use Cases of Vehicle In Use SC
        A_vehicle occupant_feel comfortable temperature
          vehicle occupant : Vehicle Occupant
          feel comfortable temperature : Feel Comfortable Temp
        Feel Comfortable Temperature
      3 System Context
«EPackage, ModelLibrary» PrimitiveTypes
«ModelLibrary» Libraries
«ModelLibrary» EcorePrimitiveTypes
```

Figure 35: *Feel Comfortable Temperature* as a child of *2 Use Cases*

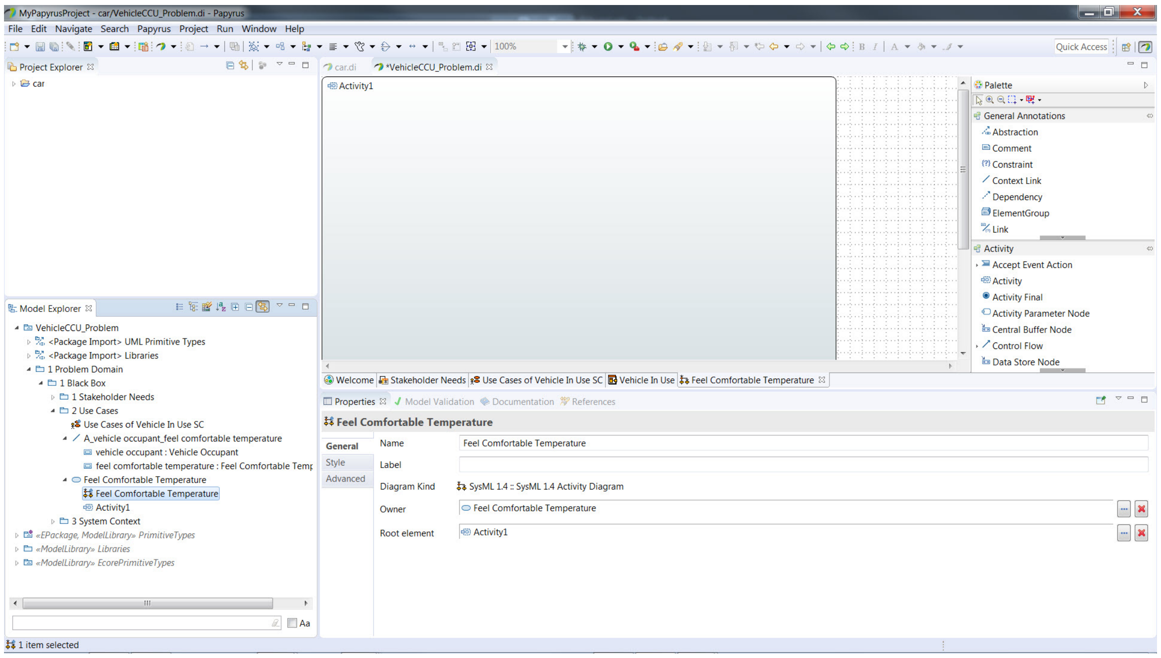
9. Now we will create an activity diagram for our use case. Right-click on the *Feel Comfortable Temperature* use case and then select **New Diagram** and then **SysML 1.4 Activity Diagram** (figure 36). Name the diagram *Feel Comfortable Temperature* (the same name as the use case). Your screen should look like figure 37. We now have the *Feel Comfortable Temperature* activity diagram as a child of the *Feel Comfortable Temperature* use case in the *Model Explorer* window.



nfortable Temperature

Name: Feel Comfortable Temperatu
Label:
Is abstract: true [✓ false]
Visibility: public
Subject: Vehicle In Use

Figure 36: Create New Activity Diagram



Feel Comfortable Temperature

Name: Feel Comfortable Temperature
Label:
Diagram Kind: SysML 1.4 :: SysML 1.4 Activity Diagram
Owner: Feel Comfortable Temperature
Root element: Activity1

Figure 37: *Feel Comfortable Temperature* Activity Diagram

10. We could start diagramming our activities in the activity diagram first and then allocate the activities to participants using swimlanes. This is a little cumbersome in Papyrus as it involves dragging the activity boxes and connectors onto the swimlanes and some effort to make your diagram “pretty” again. If we know the participants in the activity we can go ahead and create the participant swimlanes first and then populate the activities. That is the approach we will follow. In the *Model Explorer* window Right-click on the *Activity 1* then select **SysML 1.4 Child** and then **AllocateActivityPartition** (figure 38). Name this partition *Vehicle Occupant* (figure 39).

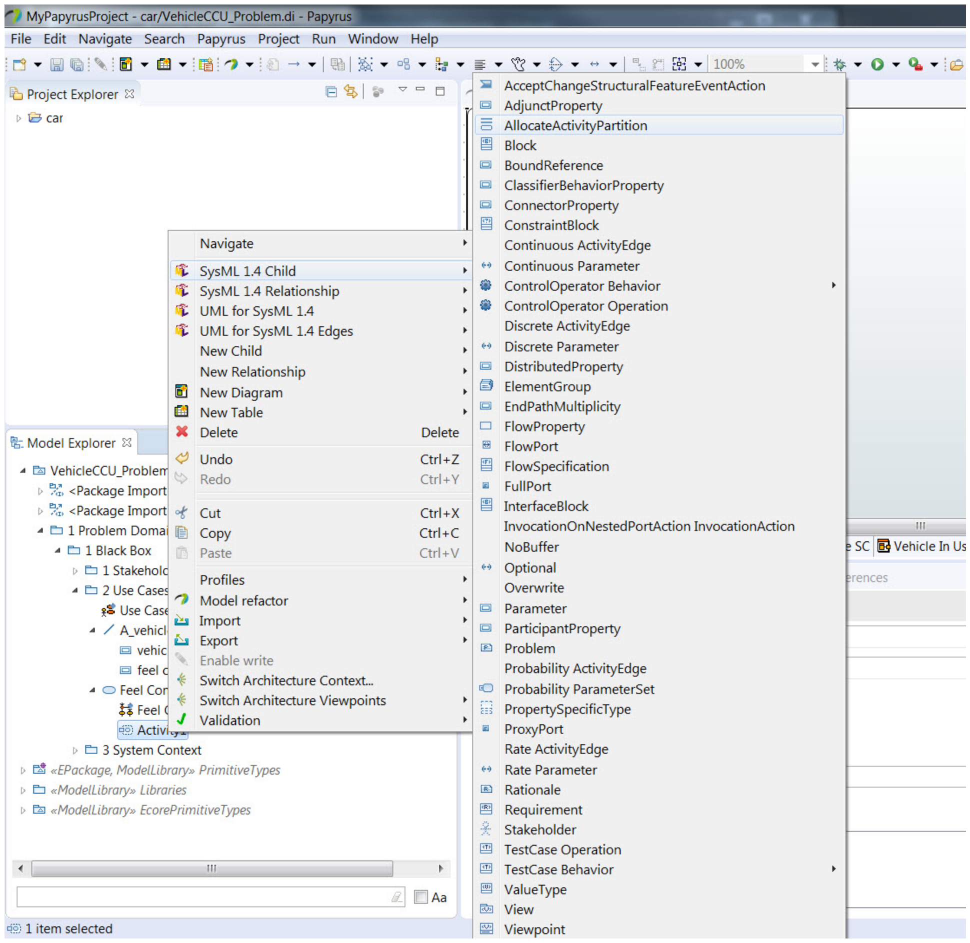
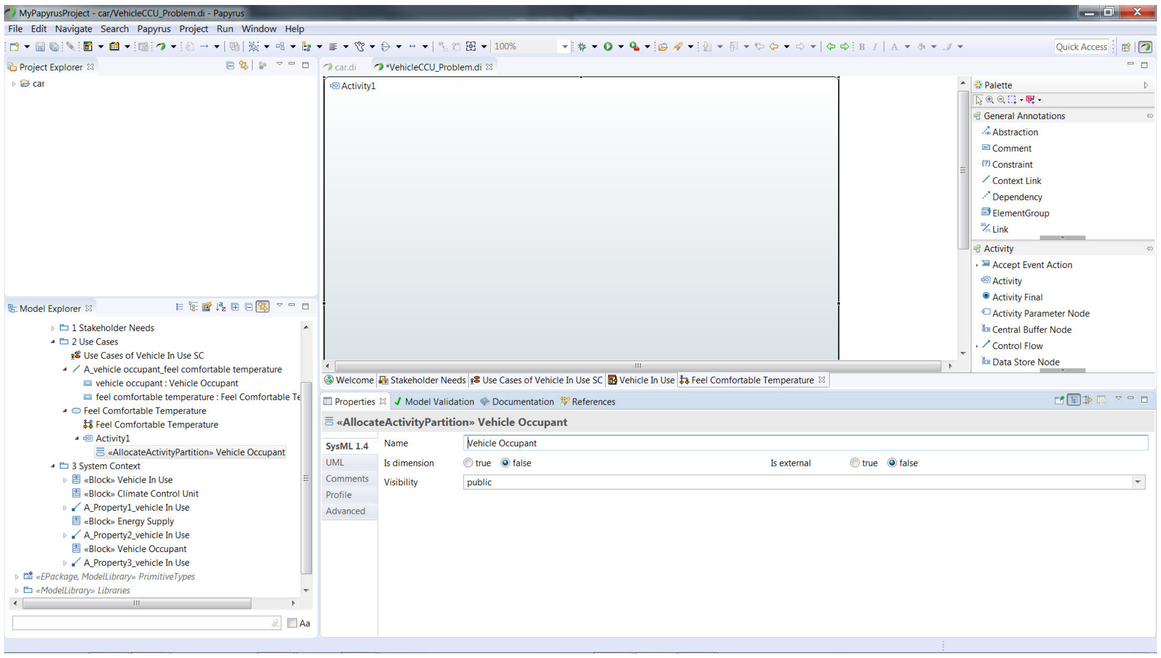


Figure 38: New Activity Partition (Swimlane)


MyPapyrusProject - car/VehicleCCU_Problem.di - Papyrus
File Edit Navigate Search Papyrus Project Run Window Help
Project Explorer
car
Model Explorer
1 Stakeholder Needs
2 Use Cases
Use Cases of Vehicle In Use SC
A_vehicle occupant_feel comfortable temperature
vehicle occupant : Vehicle Occupant
feel comfortable temperature : Feel Comfortable Te
Feel Comfortable Temperature
Feel Comfortable Temperature
Activity1
«AllocateActivityPartition» Vehicle Occupant
3 System Context
«Block» Vehicle In Use
«Block» Climate Control Unit
A_Property1_vehicle In Use
«Block» Energy Supply
A_Property2_vehicle In Use
«Block» Vehicle Occupant
A_Property3_vehicle In Use
«EPackage, ModelLibrary» PrimitiveTypes
«ModelLibrary» Libraries
car.di
*VehicleCCU_Problem.di
Activity1
Palette
General Annotations
Abstraction
Comment
Constraint
Context Link
Dependency
ElementGroup
Link
Activity
Accept Event Action
Activity
Activity Final
Activity Parameter Node
Central Buffer Node
Control Flow
Data Store Node
Welcome
Stakeholder Needs
Use Cases of Vehicle In Use SC
Vehicle In Use
Feel Comfortable Temperature
Properties
Model Validation
Documentation
References
SysML 1.4
UML
Comments
Profile
Advanced
«AllocateActivityPartition» Vehicle Occupant

Name: Vehicle Occupant
Is dimension: true [✓ false]
Is external: true [✓ false]
Visibility: public

Figure 39: Vehicle Occupant Activity Partition Creation

11. Although we have named our new activity partition *Vehicle Occupant* it does not truly represent the vehicle occupant in our use case until we type it by the *Vehicle Occupant* block. We do this as follows. First ensure that the *Vehicle Occupant* activity partition is selected in the *Model Explorer* window and you see its properties in the **Properties** tab in the bottom center of the screen (figure 39). Now click on **Advanced** tab and look for the **Represents** property and Double-click in the **Value** cell and a pop-up element selection should appear (figure 40). Expand *3 System Context* package and select the *Vehicle Occupant* and then select **OK**. The value of the **Represents** property should look like figure 41. Our activity partition is now typed by the *Vehicle Occupant* block.

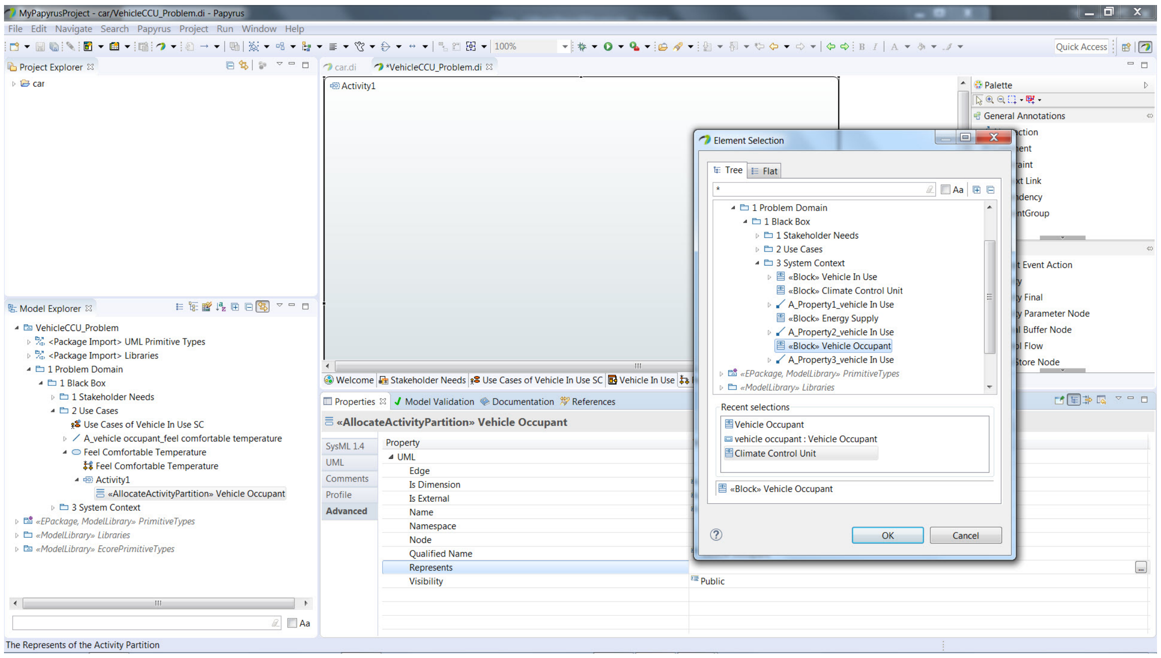
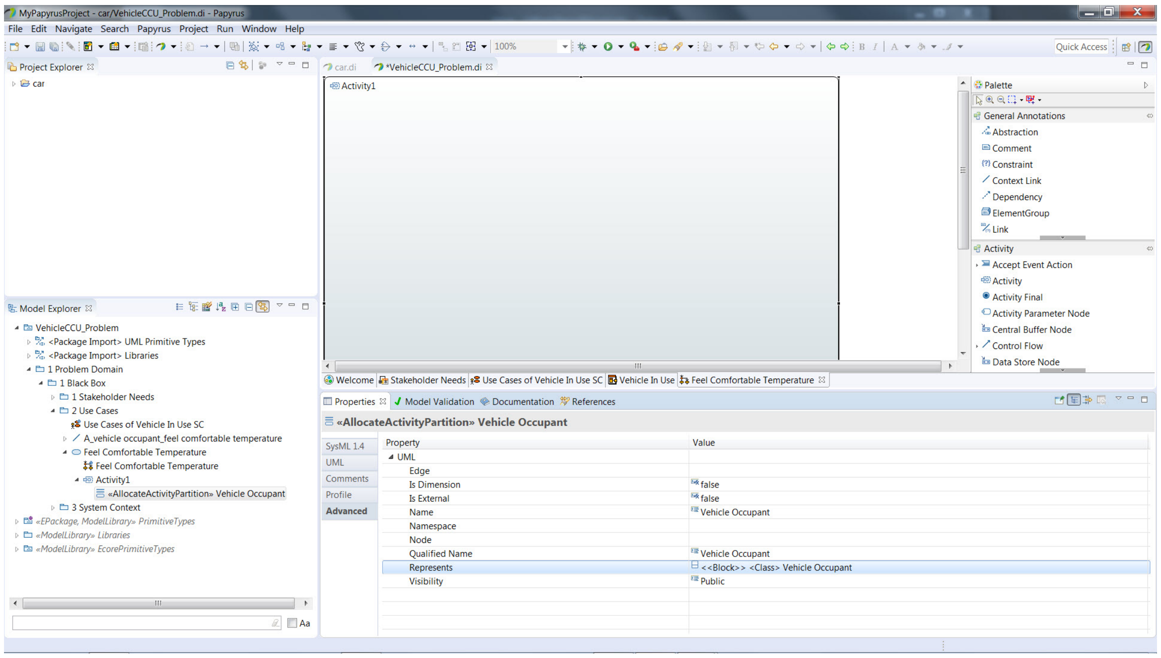


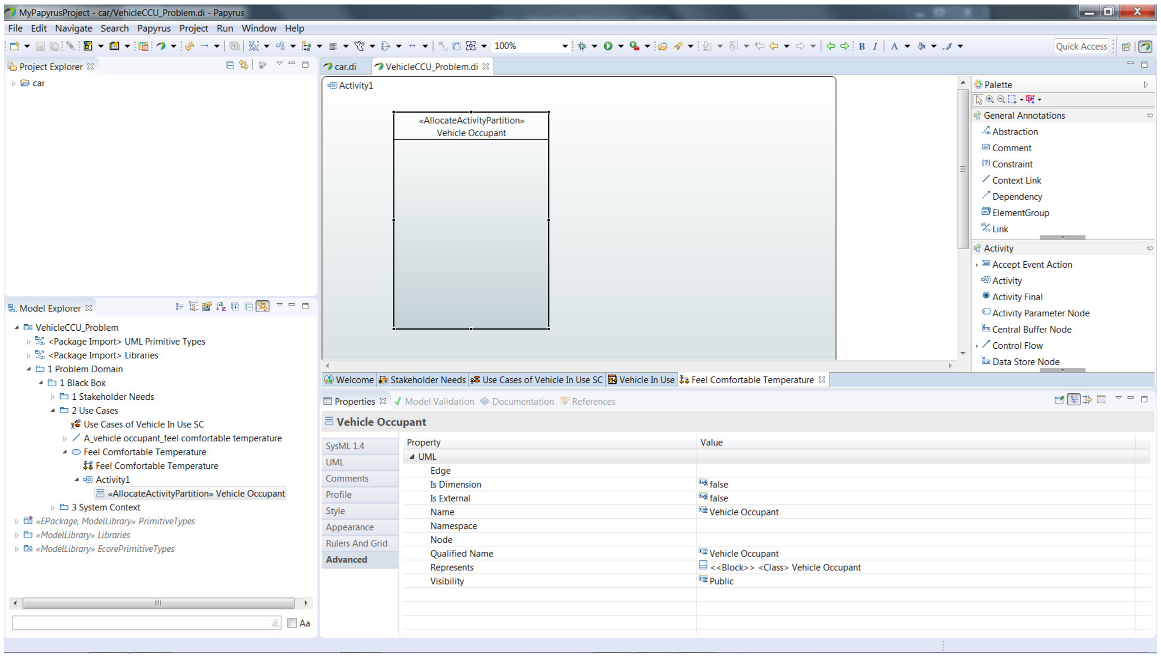
Figure 40: Vehicle Occupant Activity Partition Properties


«AllocateActivityPartition» Vehicle Occupant

Property	Value
UML	
Edge	
Is Dimension	false
Is External	false
Name	Vehicle Occupant
Namespace	
Node	
Qualified Name	Vehicle Occupant
Represents	<<Block>> <Class> Vehicle Occupant
Visibility	Public

Figure 41: Vehicle Occupant Activity Partition Typed by Vehicle Occupant Block

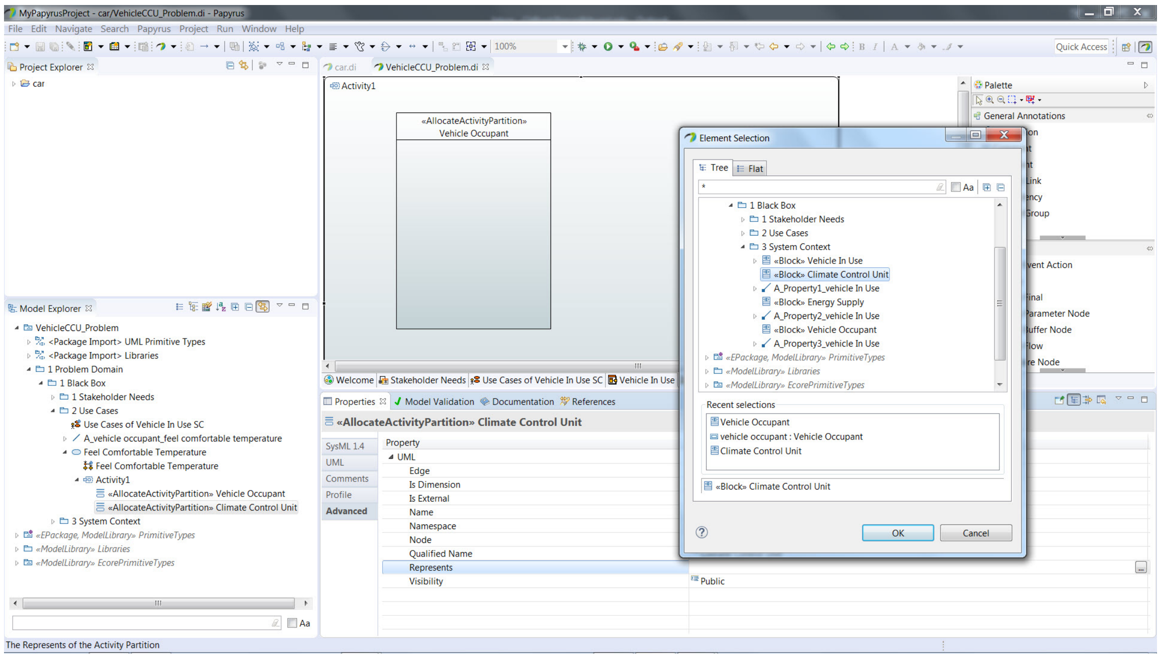
12. Now we need to add the activity partition to our activity diagram. In the *Model Explorer* window Click on the *Vehicle Occupant* activity partition that we created in the previous step and drag it to the activity diagram in the main window. You can adjust the size of the partition to make it larger (figure 42).



Property	Value
UML	
Edge	
Is Dimension	false
Is External	false
Name	Vehicle Occupant
Namespace	
Node	
Qualified Name	Vehicle Occupant
Represents	<<Block>> <Class> Vehicle Occupant
Visibility	Public

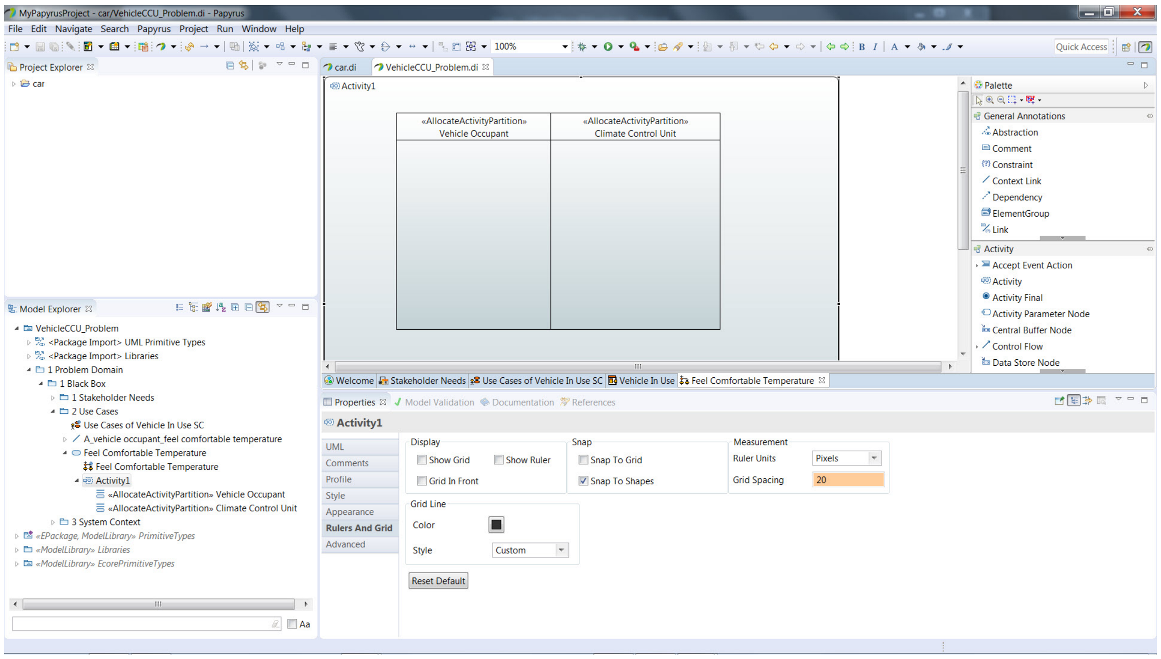
Figure 42: Vehicle Occupant Activity Partition in Activity Diagram

13. Repeat steps 10 through 12 to create another activity partition named *Climate Control Unit* and for its **Represents** property value select the *Climate Control Unit* block (figure 43). Drag this partition to your activity diagram and position next to the *Vehicle Occupant* activity partition (figure 44). To help position your shapes go to the **Rulers and Grid** tab in the **Properties** window at the bottom of the screen and un-check the **Snap to Grid** item and check the **Snap to Shapes** item.



MyPapyrusProject - car/VehicleCCU_Problem.di - Papyrus
Project Explorer
Model Explorer
Activity1
«AllocateActivityPartition»
Vehicle Occupant
Element Selection
Tree
Flat
1 Black Box
1 Stakeholder Needs
2 Use Cases
3 System Context
«Block» Vehicle In Use
«Block» Climate Control Unit
A_Property1_vehicle In Use
«Block» Energy Supply
A_Property2_vehicle In Use
«Block» Vehicle Occupant
A_Property3_vehicle In Use
Recent selections
Vehicle Occupant
vehicle occupant : Vehicle Occupant
Climate Control Unit
OK
Cancel
«AllocateActivityPartition» Climate Control Unit
Represents
The Represents of the Activity Partition

Figure 43: Create Climate Control Unit Activity Partition



MyPapyrusProject - car/VehicleCCU_Problem.di - Papyrus
Activity1
«AllocateActivityPartition»
Vehicle Occupant
«AllocateActivityPartition»
Climate Control Unit
Palette
Model Explorer
«AllocateActivityPartition» Vehicle Occupant
«AllocateActivityPartition» Climate Control Unit
Properties
Activity1
Rulers And Grid

Figure 44: Climate Control Activity Partition in Activity Diagram

14. Let’s say that our analysis of the *Feel Comfortable Temperature* use case scenario has resulted in the following actions:

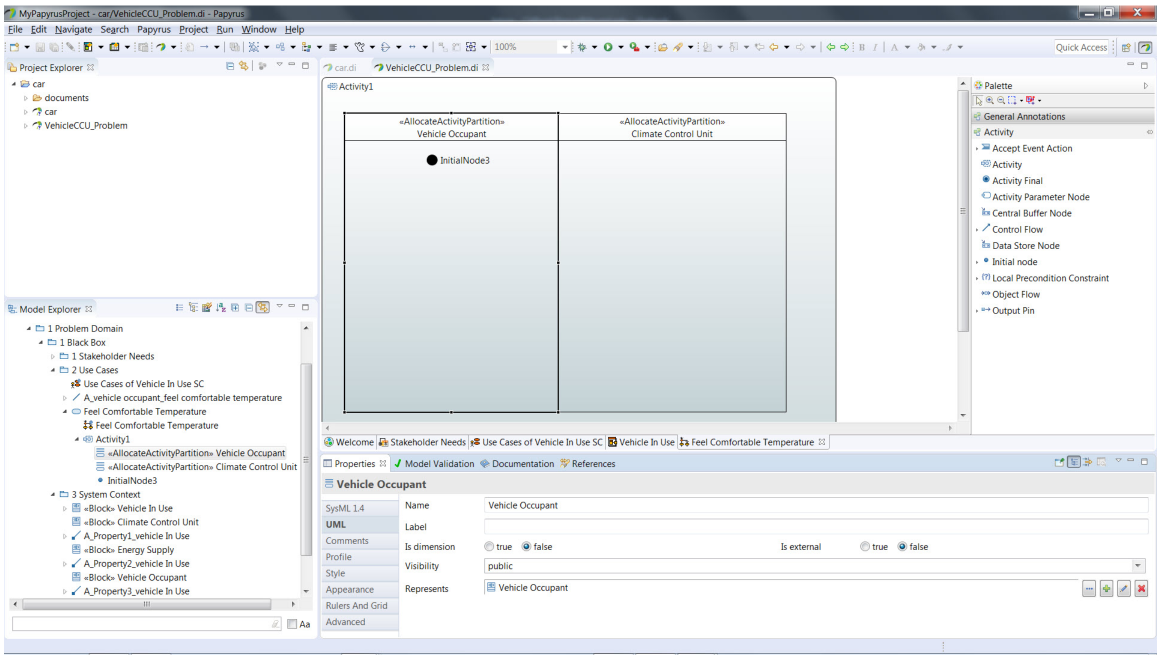
    (a) Turn on Climate Control

    (b) Check System

(c) Display Temperature
(d) Start Climate Control
(e) Set Temperature
(f) Reach Required Temperature
(g) Maintain Temperature
(h) Turn off Climate Control
(i) Stop Climate Control

Let's start creating our activity diagram.

15. Ensure that the *Feel Comfortable Temperature* activity diagram is selected in the main window. You will see the **Activity** tool palette on the right. Click on **Initial node** in the tool palette then click in the *Vehicle Occupant* activity partition and an initial node will be created (figure 45). Note that the node has a label next to it in the diagram. These labels can be cumbersome so select the *Initial node* in the activity diagram or in the *Model Explorer* window. Now select the **UML** tab in the **Properties** tab at the bottom of the screen (figure 45). Click in the **Label** field and type one blank space with the <Spacebar> and press *Enter*. The label should now be removed.



Vehicle Occupant

Name	Vehicle Occupant		
Label			
Is dimension	true [✓ false]	Is external	true [✓ false]
Visibility	public		
Represents	Vehicle Occupant		

Figure 45: Activity Diagram Initial Node

16. Now we will start creating our sequence of actions. Right-click *Activity1* in the *Model Explorer* window and select **New Child** and in the pop-up window scroll down until you find **Opaque Action** and select it (figure 46). Type **Turn On Climate Control** for the name and then press *Enter.* The *Turn On Climate Control* action now appears

as a child node under *Activity1* in the *Model Explorer* (figure 47). The action does not yet appear on our activity diagram. We will take care of that next.

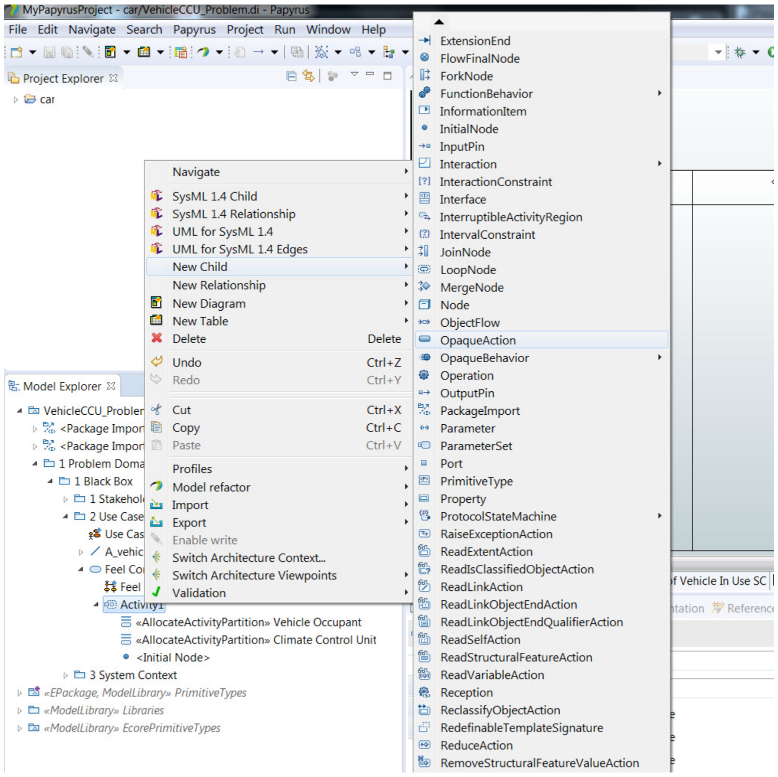
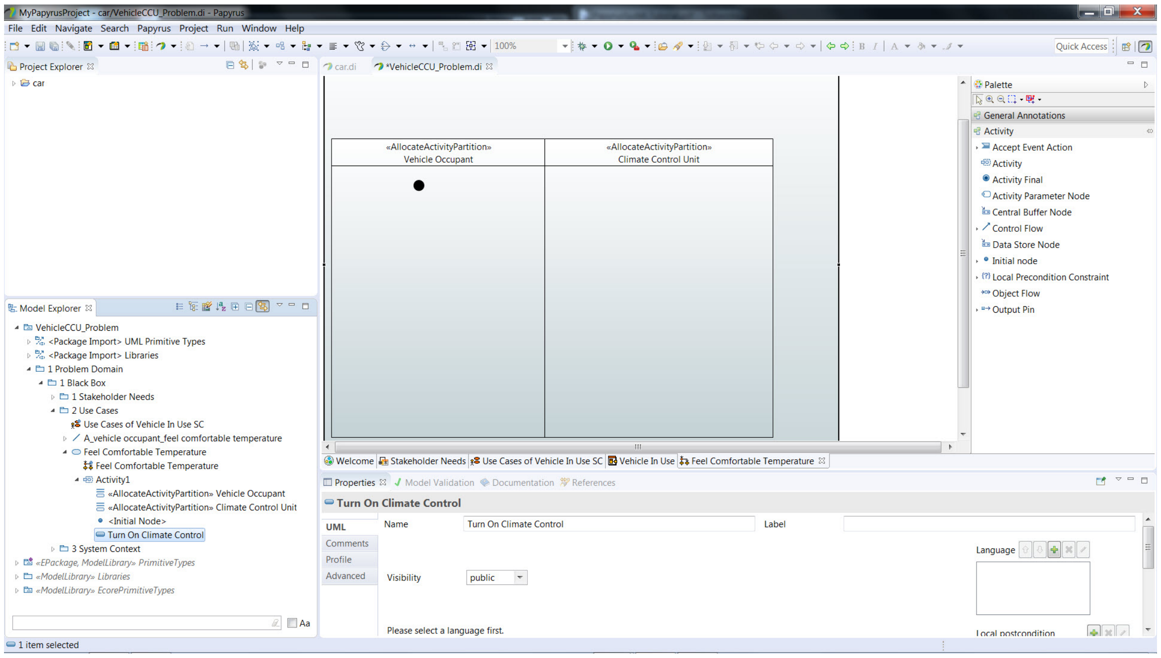


Figure 46: Create Opaque Action



MyPapyrusProject - car/VehicleCCU_Problem.di - Papyrus
File Edit Navigate Search Papyrus Project Run Window Help
100%
Quick Access
Project Explorer
car
car.di
*VehicleCCU_Problem.di
«AllocateActivityPartition»
Vehicle Occupant
«AllocateActivityPartition»
Climate Control Unit
Palette
General Annotations
Activity
Accept Event Action
Activity
Activity Final
Activity Parameter Node
Central Buffer Node
Control Flow
Data Store Node
Initial node
Local Precondition Constraint
Object Flow
Output Pin
Model Explorer
VehicleCCU_Problem
<Package Import> UML Primitive Types
<Package Import> Libraries
1 Problem Domain
1 Black Box
1 Stakeholder Needs
2 Use Cases
Use Cases of Vehicle In Use SC
A_vehicle occupant_feel comfortable temperature
Feel Comfortable Temperature
Feel Comfortable Temperature
Activity1
«AllocateActivityPartition» Vehicle Occupant
«AllocateActivityPartition» Climate Control Unit
<Initial Node>
Turn On Climate Control
3 System Context
«EPackage, ModelLibrary» PrimitiveTypes
«ModelLibrary» Libraries
«ModelLibrary» EcorePrimitiveTypes
Aa
1 item selected
Welcome
Stakeholder Needs
Use Cases of Vehicle In Use SC
Vehicle In Use
Feel Comfortable Temperature
Properties
Model Validation
Documentation
References
UML
Comments
Profile
Advanced
Language
Local postcondition

Turn On Climate Control

Name: Turn On Climate Control

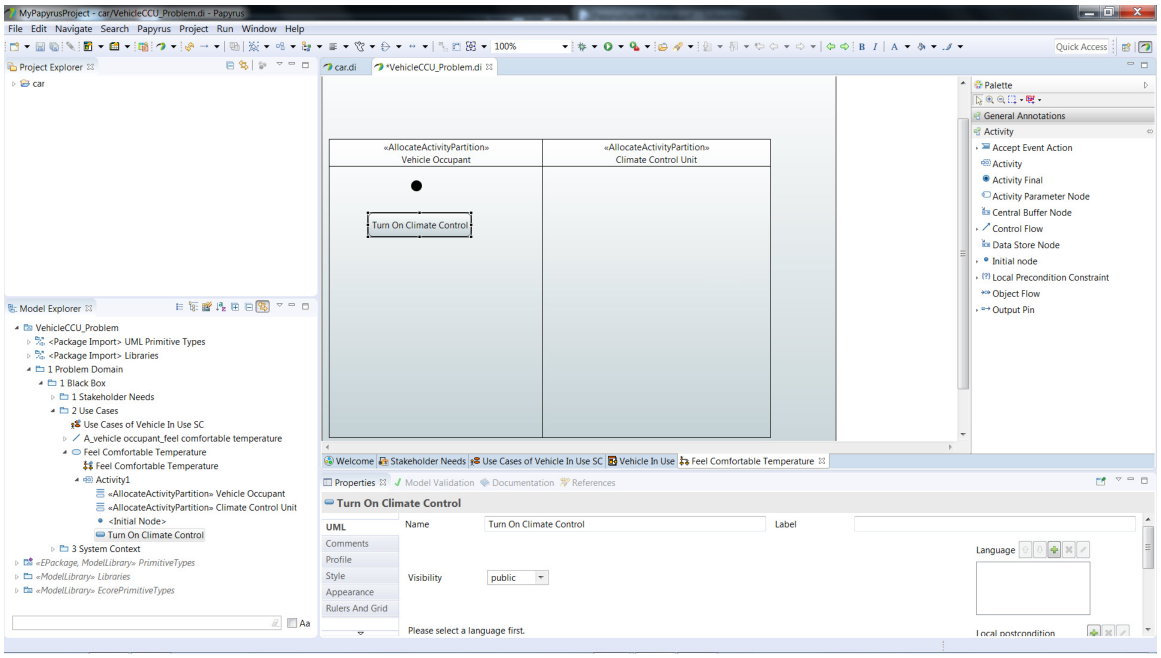
Label:

Visibility: public

Please select a language first.

Figure 47: *Turn On Climate Control* Action Created

17. Click and Drag the *Turn On Climate Control* action from the *Model Explorer* window to the *Vehicle Occupant* activity partition underneath the initial node. The *Turn On Climate Control* action should now appear on the activity diagram (figure 48).


MyPapyrusProject - car/VehicleCCU_Problem.di - Papyrus
File Edit Navigate Search Papyrus Project Run Window Help
Project Explorer
car
car.di
*VehicleCCU_Problem.di
«AllocateActivityPartition» Vehicle Occupant
«AllocateActivityPartition» Climate Control Unit
Turn On Climate Control
Palette
General Annotations
Activity
Accept Event Action
Activity
Activity Final
Activity Parameter Node
Central Buffer Node
Control Flow
Data Store Node
Initial node
Local Precondition Constraint
Object Flow
Output Pin
Model Explorer
VehicleCCU_Problem
<Package Import> UML Primitive Types
<Package Import> Libraries
1 Problem Domain
1 Black Box
1 Stakeholder Needs
2 Use Cases
Use Cases of Vehicle In Use SC
A_vehicle occupant_feel comfortable temperature
Feel Comfortable Temperature
Feel Comfortable Temperature
Activity1
«AllocateActivityPartition» Vehicle Occupant
«AllocateActivityPartition» Climate Control Unit
<Initial Node>
Turn On Climate Control
3 System Context
«EPackage, ModelLibrary» PrimitiveTypes
«ModelLibrary» Libraries
«ModelLibrary» EcorePrimitiveTypes
Welcome
Stakeholder Needs
Use Cases of Vehicle In Use SC
Vehicle In Use
Feel Comfortable Temperature
Properties
Model Validation
Documentation
References
UML
Comments
Profile
Style
Appearance
Rulers And Grid
Language
Local postcondition

Turn On Climate Control

Name: Turn On Climate Control

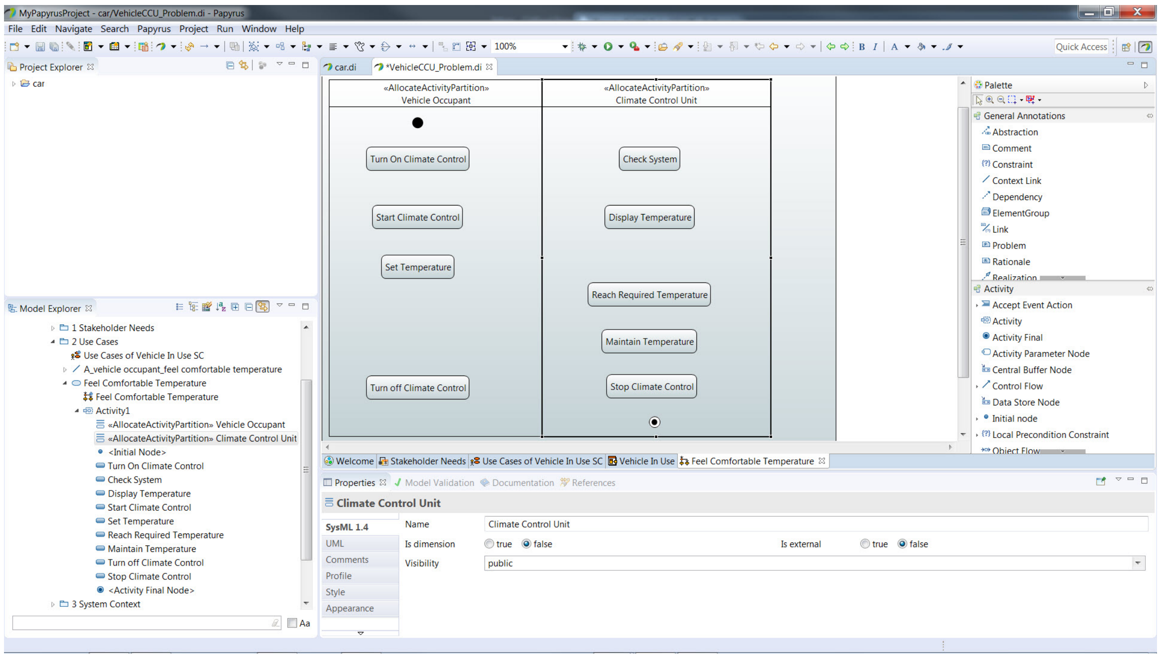
Label:

Visibility: public

Please select a language first.

Figure 48: *Turn On Climate Control* Action in Activity Diagram

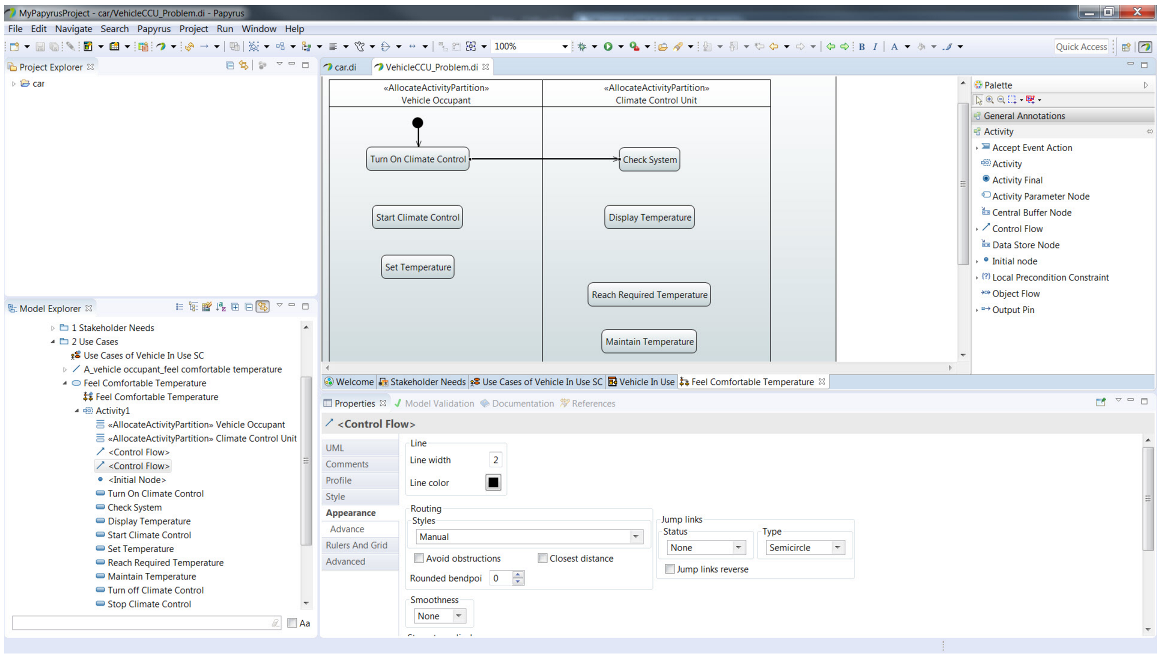
18. Repeat the procedure to create the other actions listed in step 14 and place them in the activity partitions (swimlanes). Use the activity tool palette on the right and place an **Activity Final** node underneath the *Stop Climate Control* action. Your diagram should look like (figure 49).



Name: Climate Control Unit
Is dimension: true [✓ false]
Is external: true [✓ false]
Visibility: public

Figure 49: *Feel Comfortable Temperature* Activity Diagram

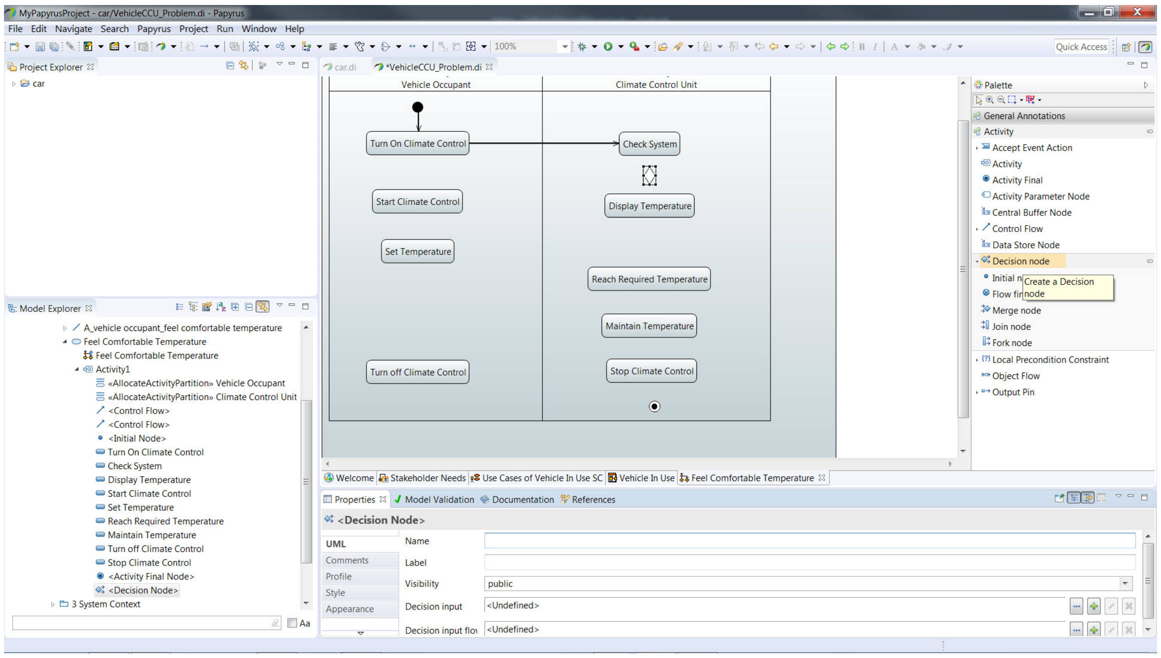
19. Now we will start adding our control flows to these actions. In the activity palette on the right Click on **Control Flow**. Now Click on the *Initial Node* in the activity diagram and then Click on the *Turn On Climate Control* action. Just hit <Enter> for the name as we will not name this control flow (I will call this an unnamed control flow from now on). A control flow will be added between these items (figure 50). If you would like the control flow line to be a little thicker you can select the **Appearance** tab in the *Properties* window and increase the value of the **Line width** parameter. Create another control flow between the *Turn On Climate Control* action and the *Check System* action (figure 50).



Line
Line width 2
Line color ■
Routing
Styles Manual
☐ Avoid obstructions ☐ Closest distance
Rounded bendpoi 0
Jump links
Status None
Type Semicircle
☐ Jump links reverse
Smoothness None

Figure 50: Control Flow

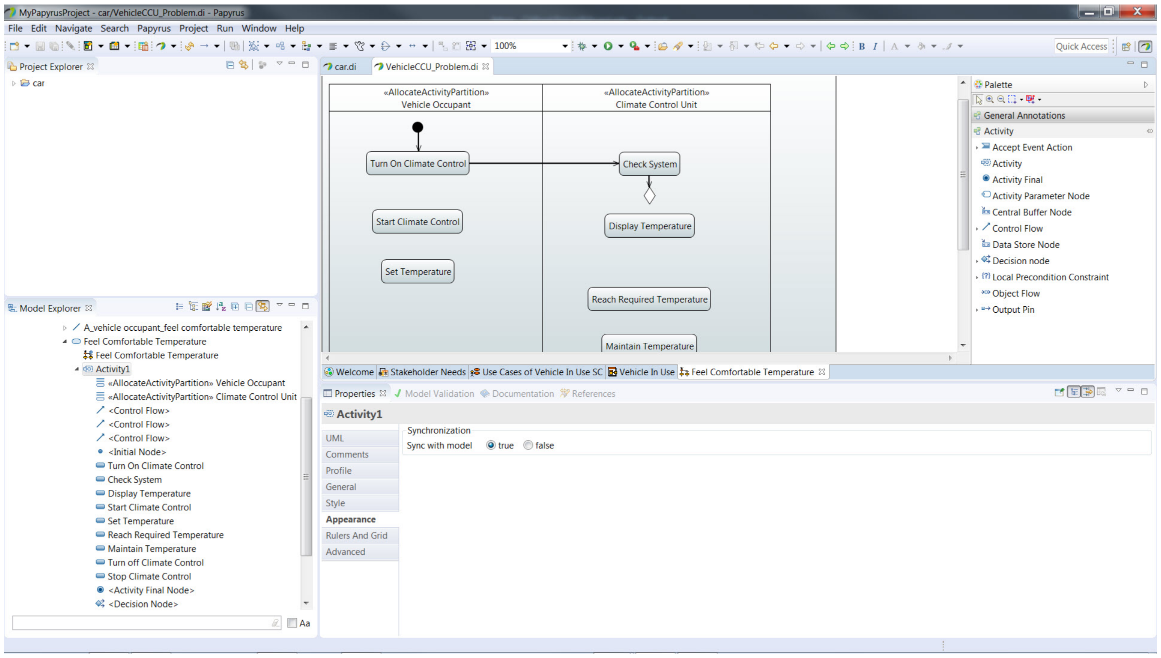
20. Now when we analyze the *Check System* action we come up with two possible cases. If the system is OK then we proceed to the next action which is *Display Temperature*, but if the system has some kind of error we want to proceed to the *Stop Climate Control* action. So, we need a decision node with a guard condition on the output of the *Display System* action. In the *Activity* palette on the right Click on the arrow next to **Initial node** to expand its contents and Click on **Decision node**. Now, in the activity diagram Click on the space between the *Check System* action and the *Display Temperature* action. A decision node will be created on the activity diagram (figure 51).



Name
Label
Visibility: public
Decision input: <Undefined>
Decision input flow: <Undefined>

Figure 51: Control Flow

21. Add a control flow between the *Check System* action and the decision node you just created (figure 52).

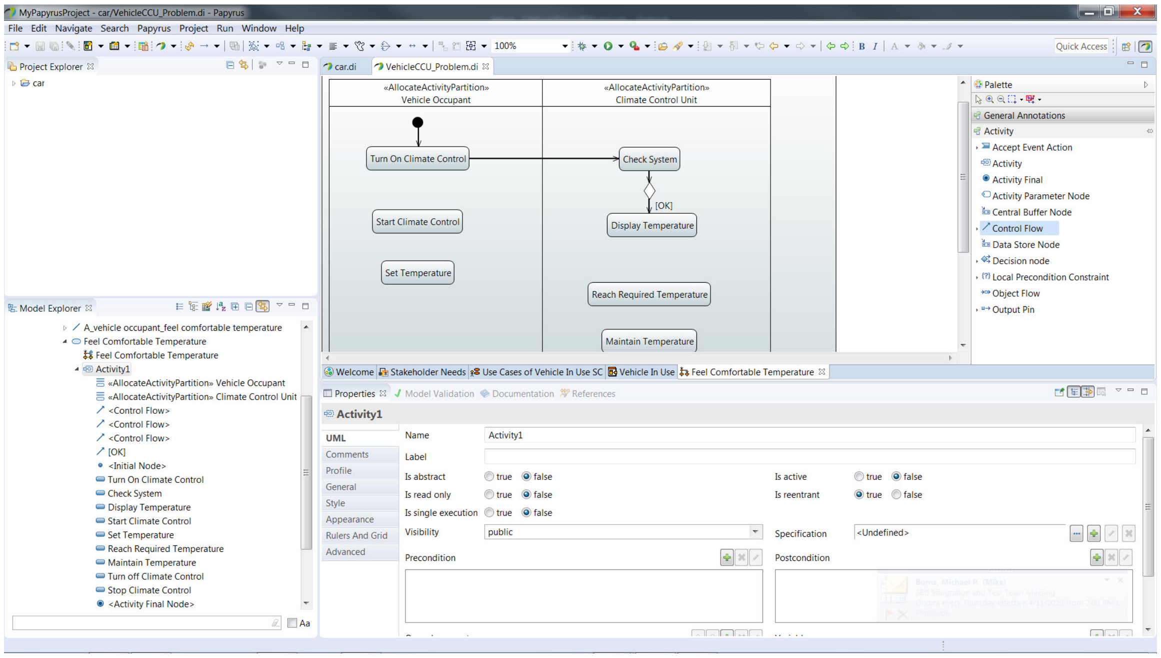

MyPapyrusProject - car/VehicleCCU_Problem.di - Papyrus
«AllocateActivityPartition» Vehicle Occupant
«AllocateActivityPartition» Climate Control Unit
Turn On Climate Control
Check System
Start Climate Control
Display Temperature
Set Temperature
Reach Required Temperature
Maintain Temperature
Palette
Properties
Activity1

Synchronization
Sync with model: [✓ true] false

Figure 52: Control Flow at Decision Node

22. Now create a control flow between the decision node and the *Display Temperature* action and type *[OK]* for the name. This represents our guard condition and therefore

this path is to be executed if the *Check System* action results show that the system is functioning normally (figure 53).

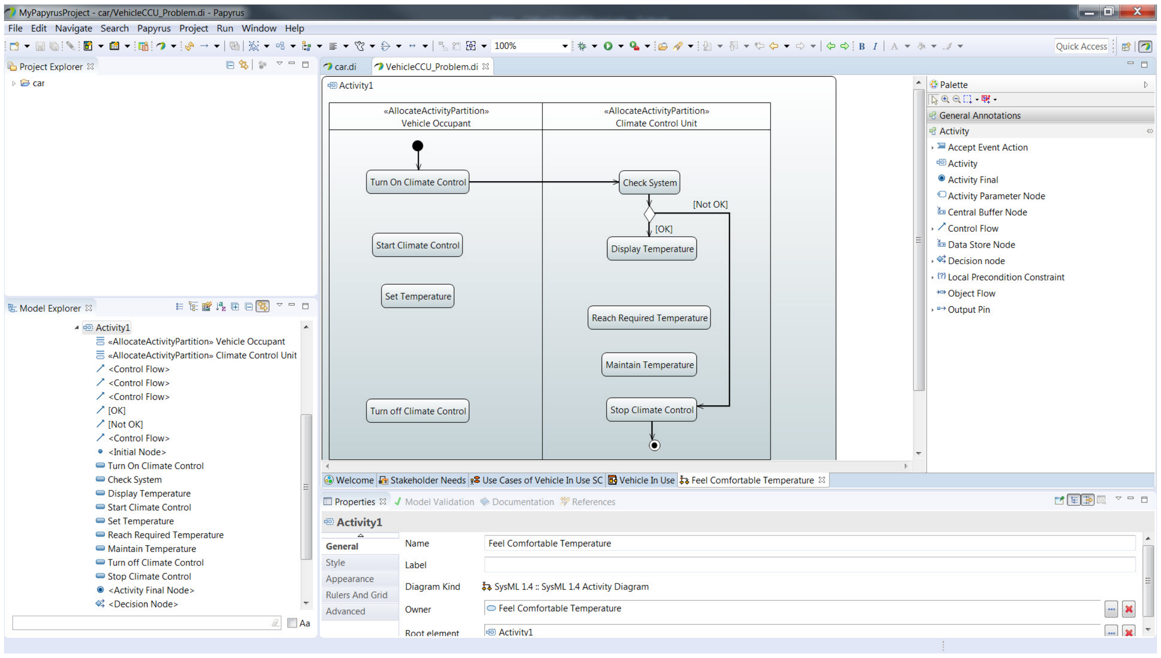


Activity1

Name: Activity1
Label:
Is abstract: true [✓ false]
Is read only: true [✓ false]
Is single execution: true [✓ false]
Visibility: public
Precondition
Is active: true [✓ false]
Is reentrant: [✓ true] false
Specification: <Undefined>
Postcondition

Figure 53: Creating the *[OK]* Guard Condition

23. Now create a control flow between the decision node and the *Stop Climate Control* action and type *Not OK* for the name. This represents our other guard condition to be executed if the *Check System* action results show that there is come kind of fault in the system. In this case we will go to the *Stop Climate Control* action. Note that you can Click on the control flow line to break it into segments and then drag these control points to reposition the segments. Add an unnamed control flow between the *Stop Climate Control* action and the final node. Your activity diagram should look like (figure 54).



Name: Feel Comfortable Temperature
Label:
Diagram Kind: SysML 1.4 :: SysML 1.4 Activity Diagram
Owner: Feel Comfortable Temperature
Root element: Activity1

Figure 54: Creating the *[Not OK]* Guard Condition

24. The next action after displaying the temperature is to start the climate control so create an unnamed control flow from the *Display Temperature* action to the *Start Climate Control* action. After the climate control is started the next action is to set the temperature so create an unnamed control flow between the *Start Climate Control* action and the *Set Temperature* action. Add an unnamed control flow between the *Set Temperature* action and the *Reach Required Temperature* action. Add an unnamed control flow between the *Turn off Climate Control* action and the *Stop Climate Control* action. Your activity diagram should look like (figure 55).

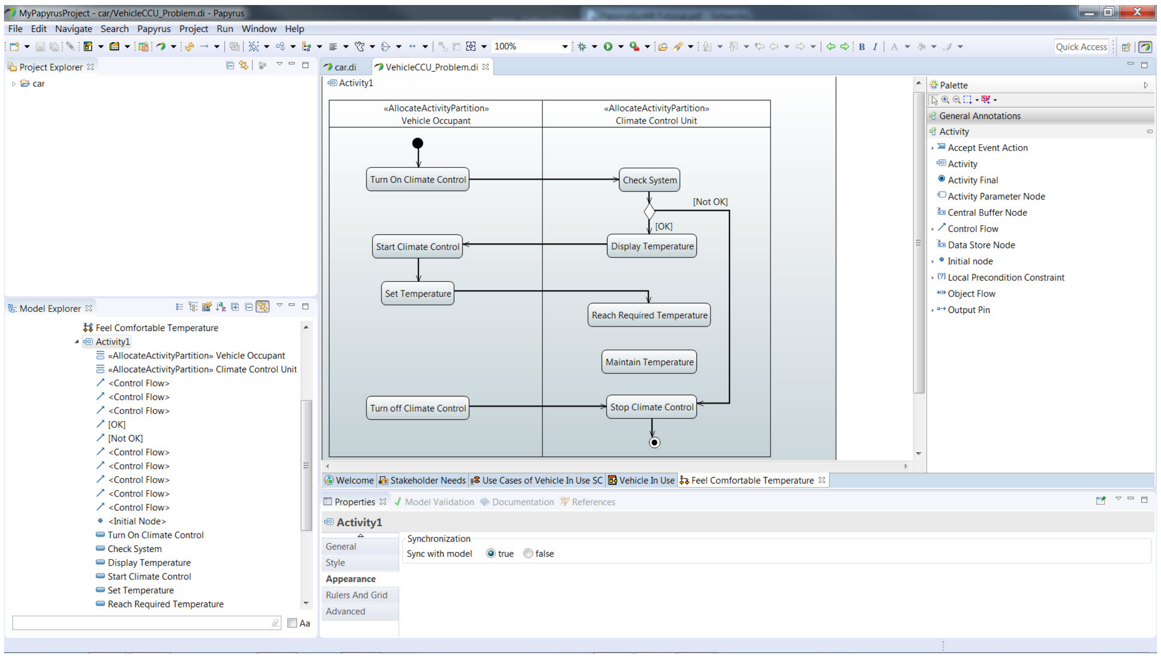


Figure 55: Creating More Control Flows

25. To complete our control flows for this simple example we need to add a couple more decision nodes; One between the *Reach Required Temperature* action and the *Maintain Temperature* action and another below the *Maintain Temperature* action. We also need to add the remaining control flows. Use your skills from the previous steps to make your diagram look like (figure 56).

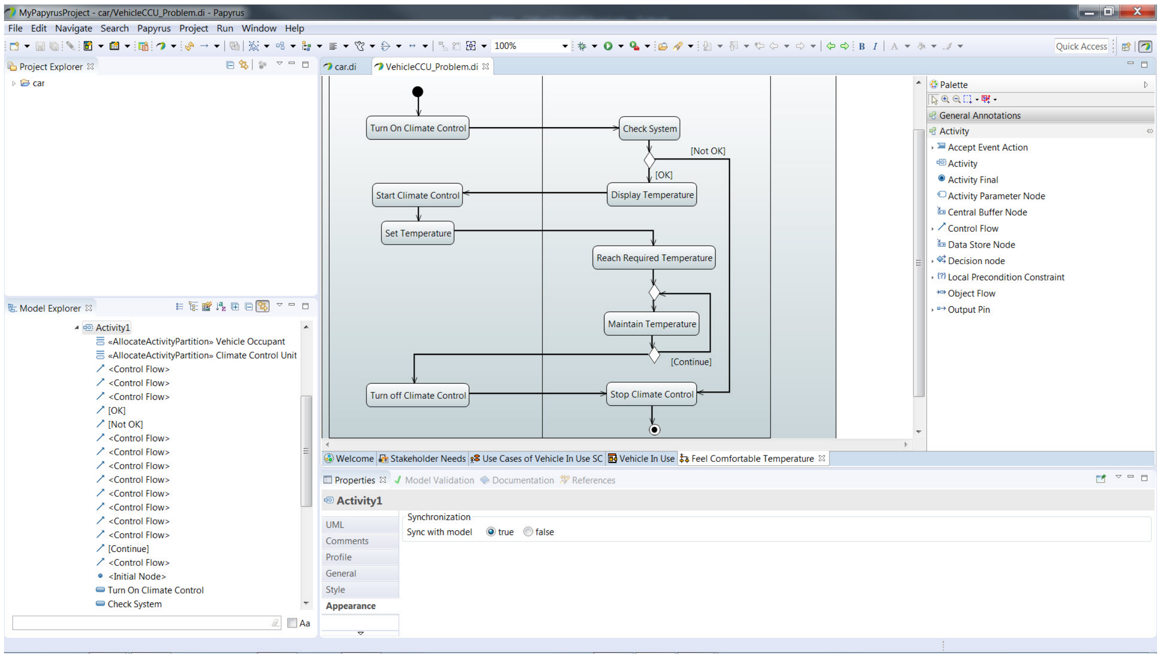


Figure 56: Final Control Flows

26. Now we have the actions and the control flow in our activity diagram but we need to pass some data as well. The required temperature should be an output from the *Set Temperature* action and this is provided as an input to both the *Reach Required Temperature* action and the *Maintain Temperature* action. In this way the system knows the desired temperature that it is to reach and maintain once it is set by the user. First create a fork node by Clicking on **Fork node** from the *Activity* palette and then Click in the space underneath the *Set Temperature* action. Size the fork as shown in (figure 57).

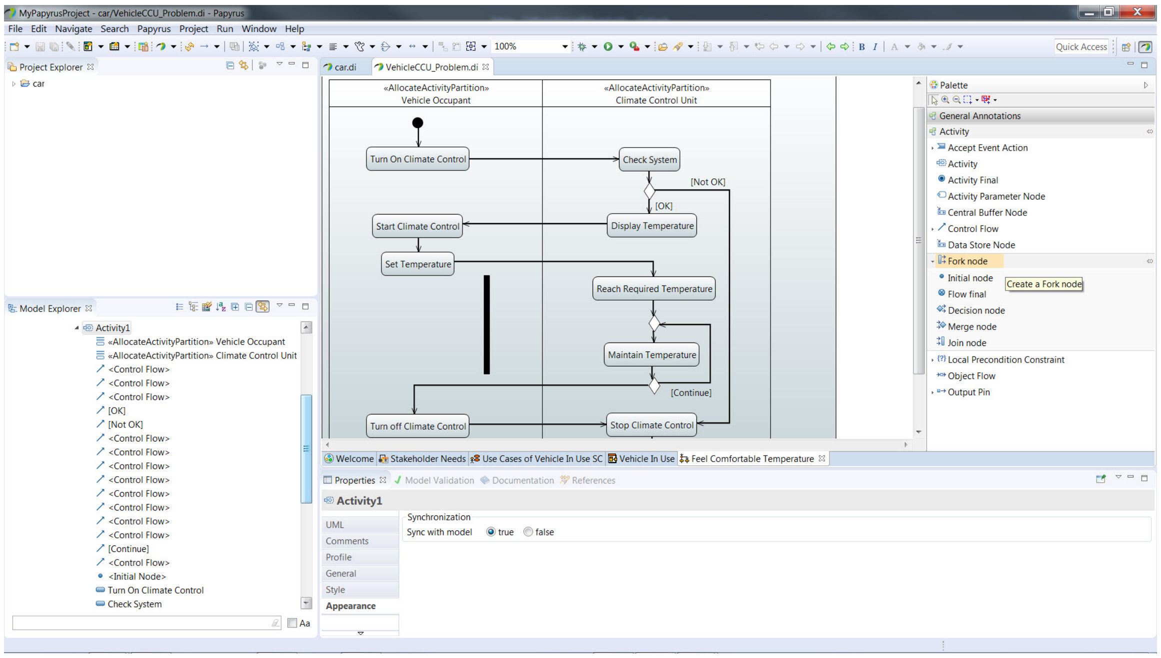
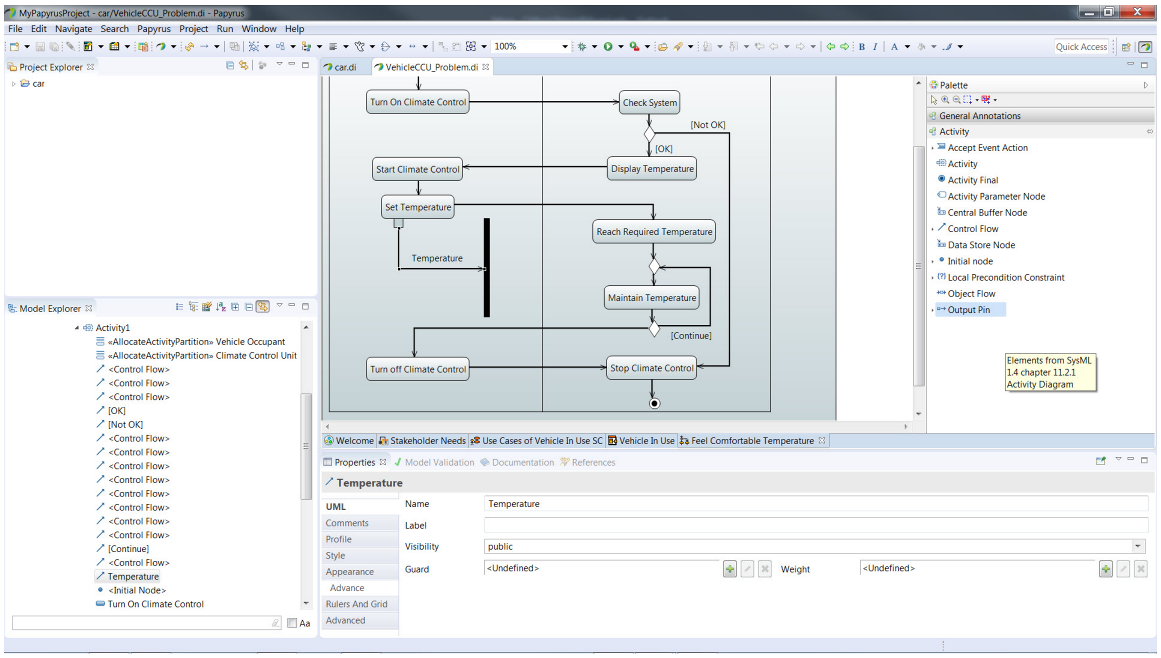


Figure 57: Create Fork Node

27. Now we will create an output pin on the *Set Temperature* action to pass the *Temperature* as an output. In the *Activity* palette on the right find the **Output Pin** item and Click on it. Now in the activity diagram Click on the *Set Temperature* action and the output pin will be attached to the action. You can Click and Drag the output pin to the bottom of the *Set Temperature* action. Now create a control flow from the output pin to the fork node and name the control flow *Temperature*. Your activity diagram should look like (figure 58).



Temperature

Name: Temperature
Label:
Visibility: public
Guard: <Undefined>
Weight: <Undefined>

Figure 58: Temperature Output

28. Now we need to provide the *Temperature* as input to both the *Reach Required Temperature* action and the *Maintain Temperature* action. To do this go to the *Activity* palette on the right and Click on **Input Pin** then Click on the *Reach Required Temperature* action and an input pin will be created. Position this pin on the left side of the action if not already there. Repeat the procedure to create an input pin on the *Maintain Temperature* action. Now create control flows between the fork node and each input pin you just created. Your activity diagram should look like (figure 59) and our example activity diagram is complete.

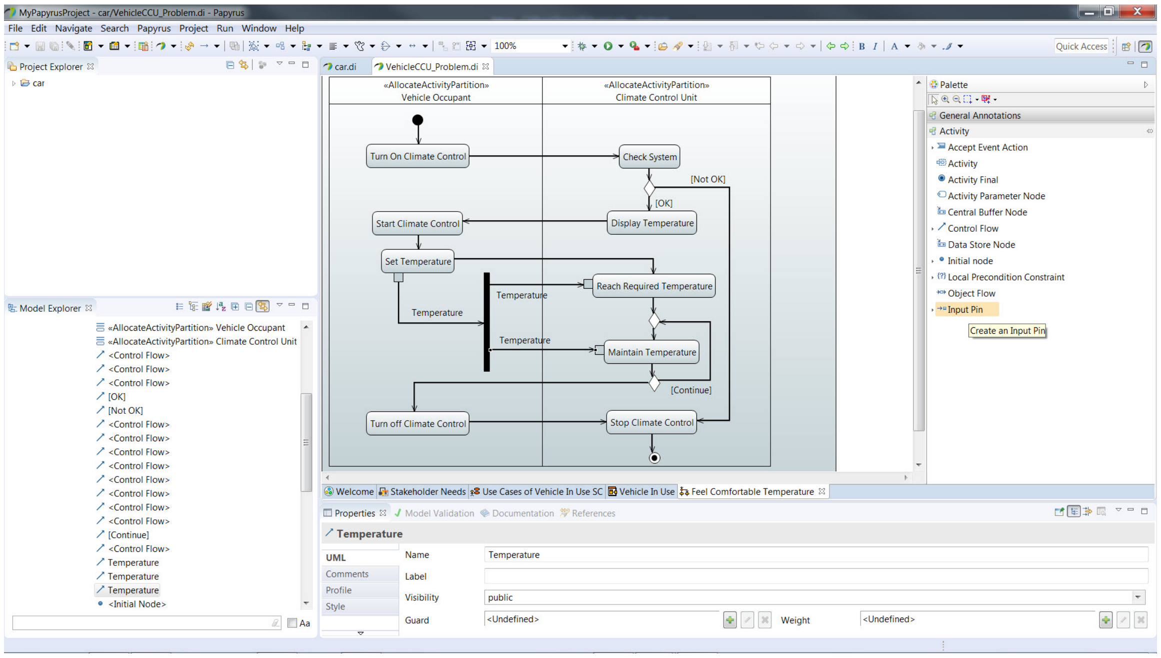
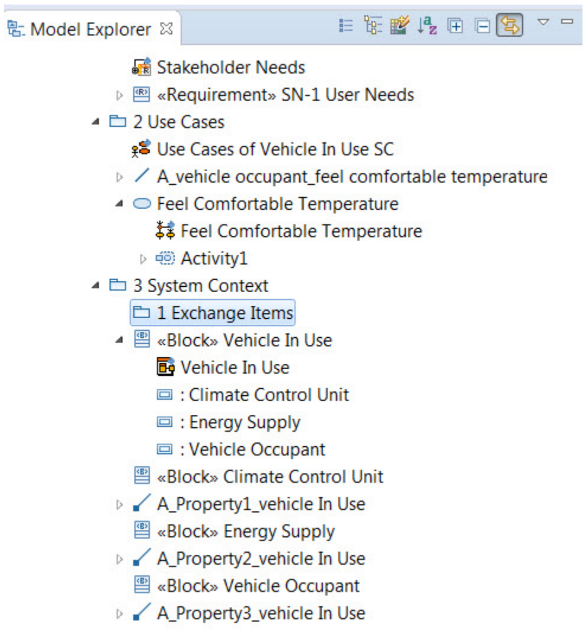


Figure 59: Temperature Input

2.2.6 Step 6. Finalize *Vehicle In Use* Internal Block Diagram

We now analyze the final activity diagram in (figure 59) and consider the flow of objects and control across the activity partition (swimlane) and we see that there is a flow of control as the vehicle occupant controls the climate control system and there is a flow of status information about the system such as the temperature. So now let's go back and update our *Vehicle In Use* internal block diagram and add these flows. First, however, let's create a package to hold the *Control* and *Status* flows that we will use in the diagram.

1. By now you should be familiar with how to create a new package in the *Model Explorer* window. Create a new child package in the *3 Systems Context* package and name the package *1 Exchange Items* (figure 60).



```
Model Explorer
      Stakeholder Needs
    ▷ «Requirement» SN-1 User Needs
  ◢ 2 Use Cases
      Use Cases of Vehicle In Use SC
    ▷ A_vehicle occupant_feel comfortable temperature
    ◢ Feel Comfortable Temperature
        Feel Comfortable Temperature
      ▷ Activity1
  ◢ 3 System Context
      1 Exchange Items
    ◢ «Block» Vehicle In Use
        Vehicle In Use
        : Climate Control Unit
        : Energy Supply
        : Vehicle Occupant
      «Block» Climate Control Unit
    ▷ A_Property1_vehicle In Use
      «Block» Energy Supply
    ▷ A_Property2_vehicle In Use
      «Block» Vehicle Occupant
    ▷ A_Property3_vehicle In Use
```

Figure 60: *Exchange Items* package

2. Now we will create a new signal. Right Click on the *1 Exchange Items* package that you just created and select **UML for SysML 1.4** and then **Signal** (figure 61). Type **Control** for the name and your *Model Explorer* window should look like (figure 62).

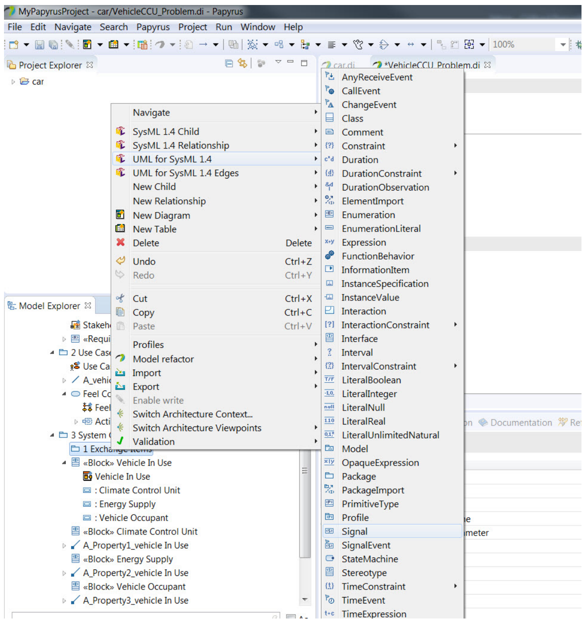
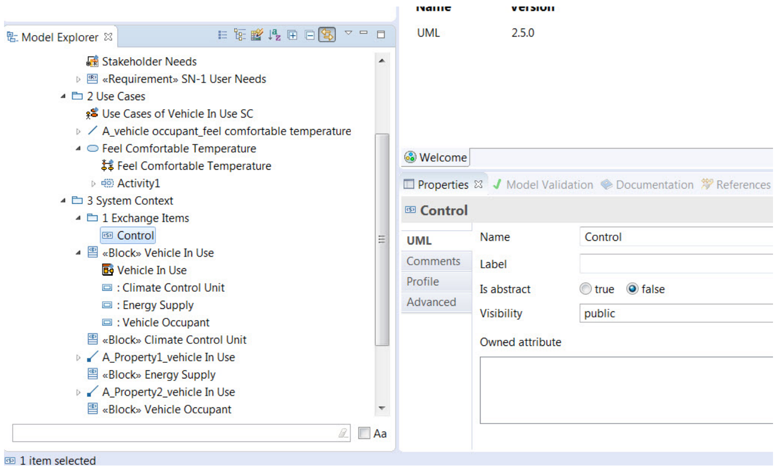


Figure 61: Create New Signal



```
Model Explorer
    Stakeholder Needs
  ▷ «Requirement» SN-1 User Needs
◢ 2 Use Cases
    Use Cases of Vehicle In Use SC
  ▷ A_vehicle occupant_feel comfortable temperature
  ◢ Feel Comfortable Temperature
      Feel Comfortable Temperature
    ▷ Activity1
◢ 3 System Context
  ◢ 1 Exchange Items
      Control
  ◢ «Block» Vehicle In Use
      Vehicle In Use
      : Climate Control Unit
      : Energy Supply
      : Vehicle Occupant
    «Block» Climate Control Unit
  ▷ A_Property1_vehicle In Use
    «Block» Energy Supply
  ▷ A_Property2_vehicle In Use
    «Block» Vehicle Occupant
1 item selected
```

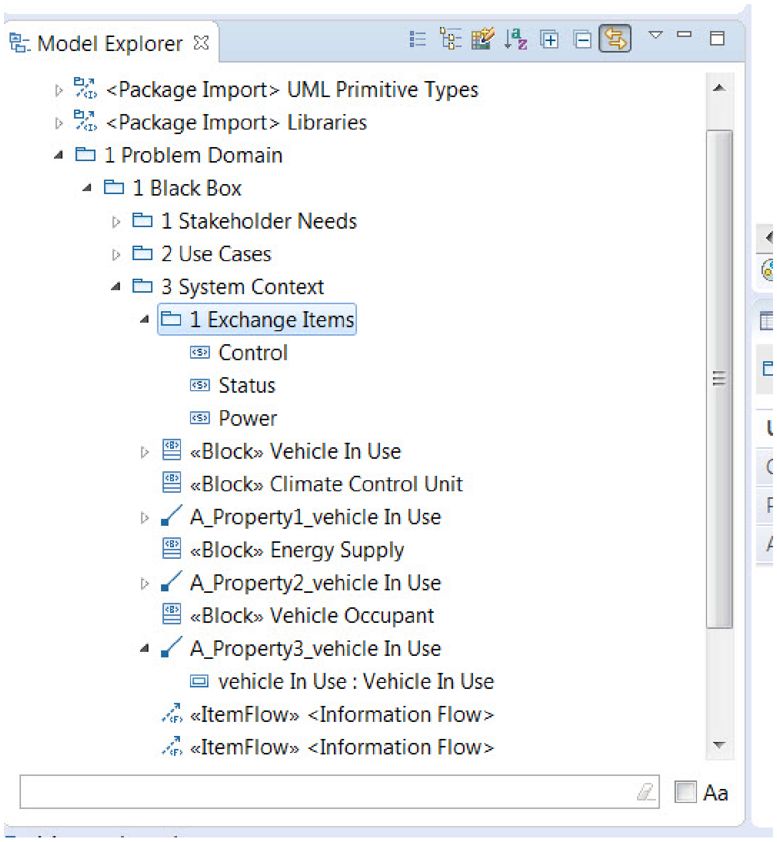
Name	Version
UML	2.5.0

Properties — Control (UML)

Name	Control
Label	
Is abstract	true [✓ false]
Visibility	public
Owned attribute	

Figure 62: *Control* Signal Created

3. Follow the same procedure to create another signal called *Status* (figure 63) and another called *Power*.

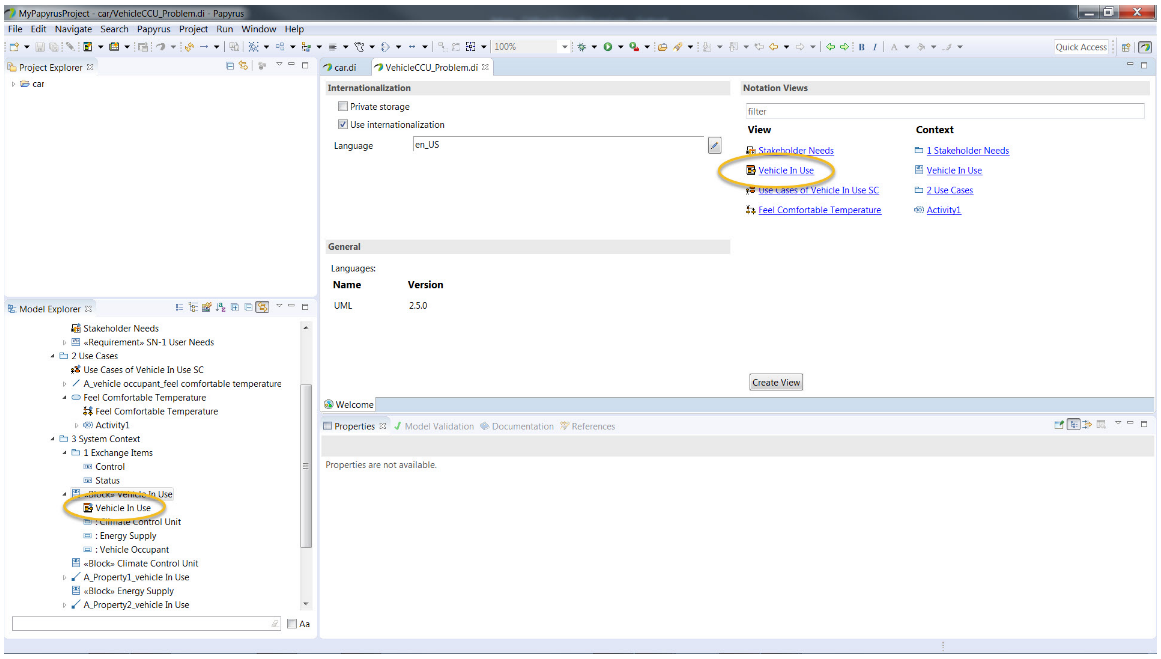


```
Model Explorer
▷ <Package Import> UML Primitive Types
▷ <Package Import> Libraries
◢ 1 Problem Domain
  ◢ 1 Black Box
    ▷ 1 Stakeholder Needs
    ▷ 2 Use Cases
    ◢ 3 System Context
      ◢ 1 Exchange Items
          Control
          Status
          Power
      ▷ «Block» Vehicle In Use
        «Block» Climate Control Unit
      ▷ A_Property1_vehicle In Use
        «Block» Energy Supply
      ▷ A_Property2_vehicle In Use
        «Block» Vehicle Occupant
      ◢ A_Property3_vehicle In Use
          vehicle In Use : Vehicle In Use
        «ItemFlow» <Information Flow>
        «ItemFlow» <Information Flow>
```

Figure 63: Signals Created

4. Open the *Vehicle In Use* internal block diagram by either Double-clicking the diagram name in the *Model Explorer* OR selecting the diagram from the *Welcome* tab in the

main window (figure 64). Once the diagram is open your screen should look like (figure 65).



Internationalization

- Private storage
- [✓] Use internationalization

Language: en_US

Notation Views

filter

View	Context
Stakeholder Needs	1 Stakeholder Needs
Vehicle In Use	Vehicle In Use
Use Cases of Vehicle In Use SC	2 Use Cases
Feel Comfortable Temperature	Activity1

General

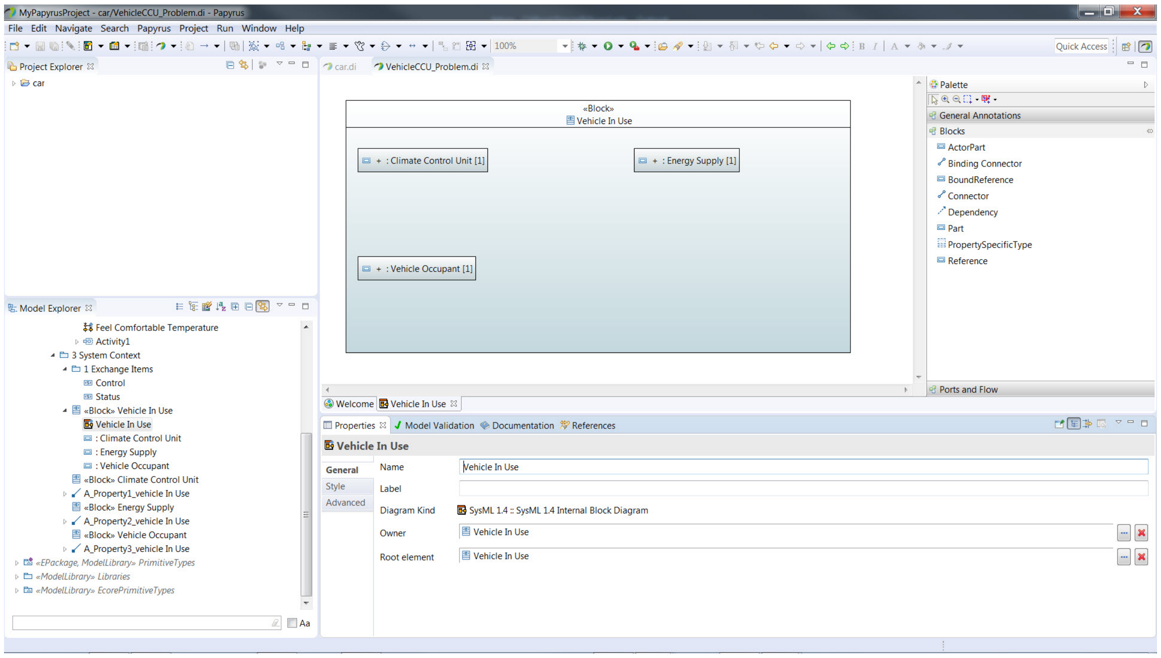
Languages:

Name	Version
UML	2.5.0

Properties are not available.

- Stakeholder Needs
- «Requirement» SN-1 User Needs
- 2 Use Cases
  - Use Cases of Vehicle In Use SC
  - A_vehicle occupant_feel comfortable temperature
  - Feel Comfortable Temperature
    - Feel Comfortable Temperature
    - Activity1
- 3 System Context
  - 1 Exchange Items
    - Control
    - Status
  - «Block» Vehicle In Use
    - Vehicle In Use
    - : Climate Control Unit
    - : Energy Supply
    - : Vehicle Occupant
  - «Block» Climate Control Unit
  - A_Property1_vehicle In Use
  - «Block» Energy Supply
  - A_Property2_vehicle In Use

Figure 64: Open *Vehicle In Use* Internal Block Diagram



- Feel Comfortable Temperature
- Activity1
- 3 System Context
  - 1 Exchange Items
    - Control
    - Status
  - «Block» Vehicle In Use
    - Vehicle In Use
    - : Climate Control Unit
    - : Energy Supply
    - : Vehicle Occupant
  - «Block» Climate Control Unit
  - A_Property1_vehicle In Use
  - «Block» Energy Supply
  - A_Property2_vehicle In Use
  - «Block» Vehicle Occupant
  - A_Property3_vehicle In Use
- «EPackage, ModelLibrary» PrimitiveTypes
- «ModelLibrary» Libraries
- «ModelLibrary» EcorePrimitiveTypes

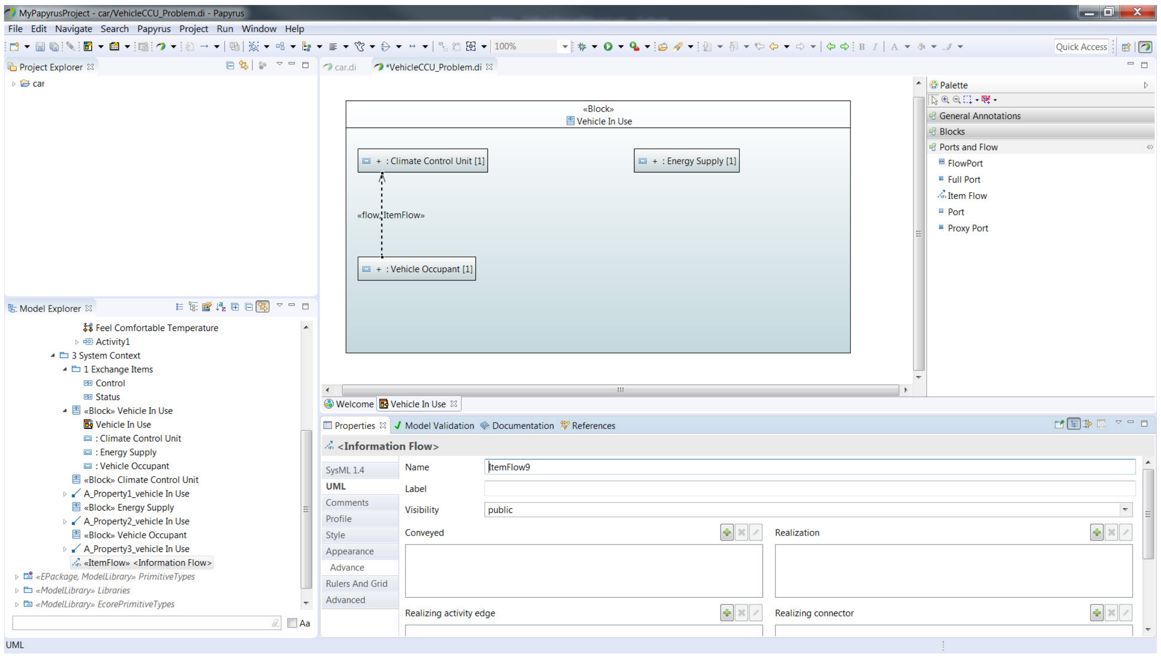
Vehicle In Use

Name	Vehicle In Use
Label	
Diagram Kind	SysML 1.4 :: SysML 1.4 Internal Block Diagram
Owner	Vehicle In Use
Root element	Vehicle In Use

Figure 65: *Vehicle In Use* Internal Block Diagram

5. Now we will create the item flow representing our *Control* signal. In the **Ports and Flow** palette on the right Click on **Item Flow**. Now in internal block diagram Click on

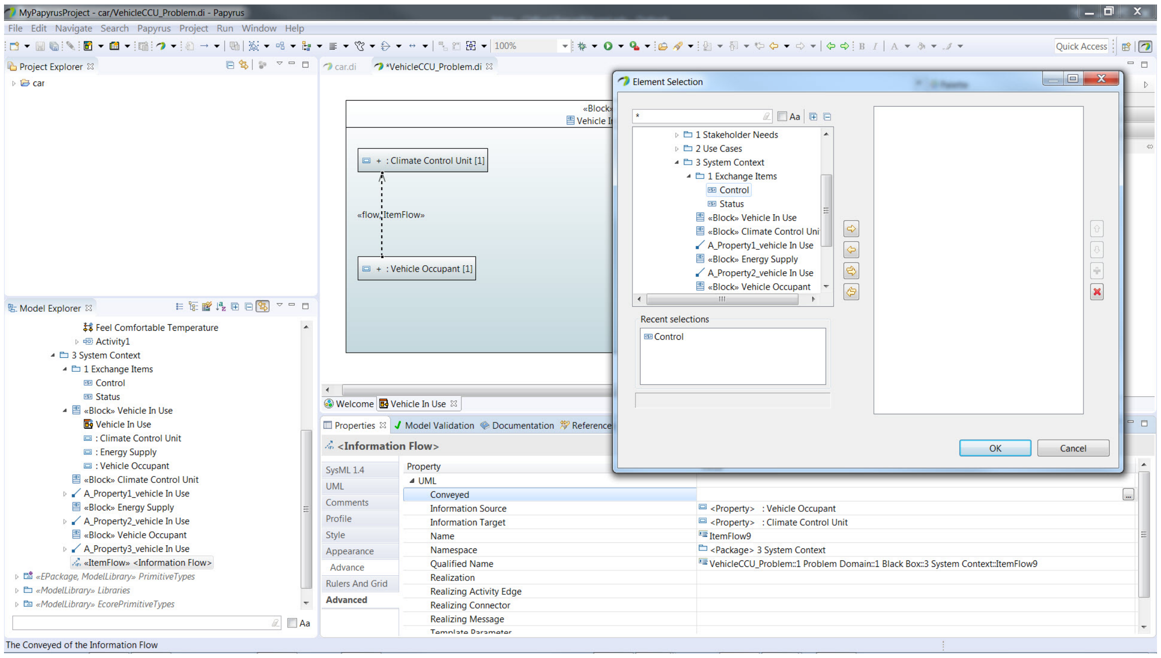
the *Vehicle Occupant* block and then on the *Climate Control Unit* block. The item flow is created (figure 66). Note that you can type a <space> in the label field if you don't want to see the name in the diagram.



Name: ItemFlow9
Label:
Visibility: public

Figure 66: *Vehicle In Use* Internal Block Diagram

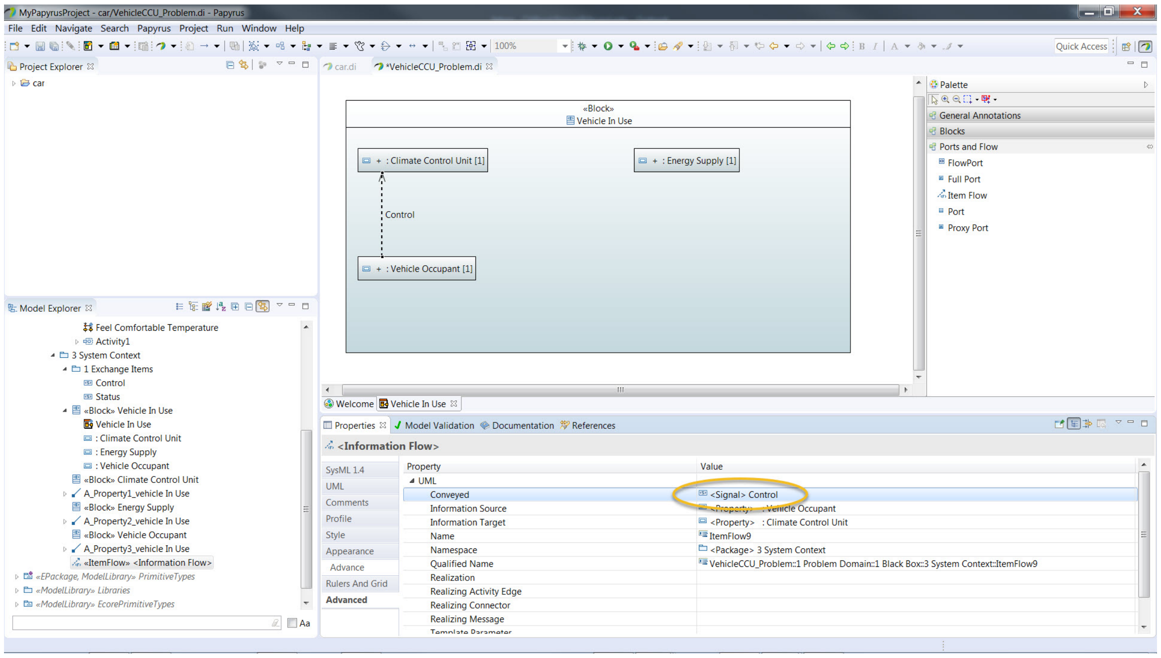
6. Ensure that the item flow that you just created is selected and in the **Properties** window at the bottom of the screen select the **Advanced** tab. Double-click in the value cell of the **Conveyed** property to open the **Element Selection** window. Expand the packages until you see the *1 Exchange Items* package created earlier. Double-click on the *Control* signal in this package and then click **OK** (figure 67).



Property	Value
UML	
Conveyed	
Information Source	<Property> : Vehicle Occupant
Information Target	<Property> : Climate Control Unit
Name	ItemFlow9
Namespace	<Package> 3 System Context
Qualified Name	VehicleCCU_Problem::1 Problem Domain::1 Black Box::3 System Context::ItemFlow9
Realization	
Realizing Activity Edge	
Realizing Connector	
Realizing Message	

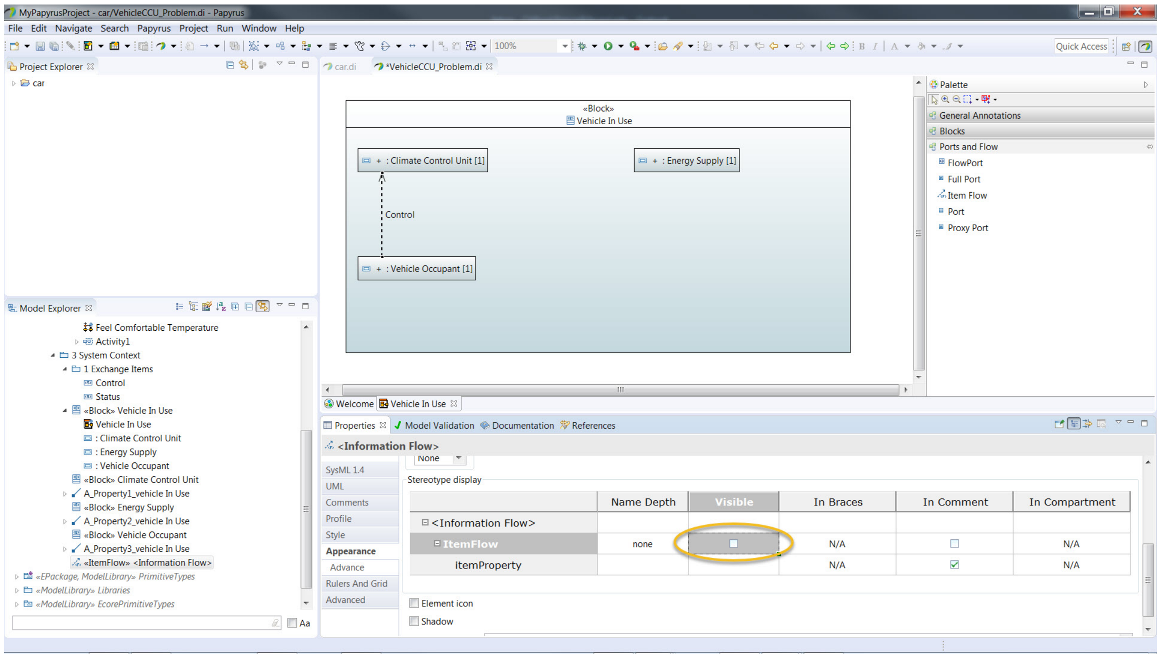
Figure 67: Conveying the *Control* Signal

7. Notice that the value of the **Conveyed** property now shows our *Control* signal (figure 68) and it appears on our item flow in the internal block diagram. Note that to eliminate the other labels you can go the **Appearance** tab and uncheck the item at the intersection of **ItemFlow** and **Visible** (figure 69).



Property	Value
UML	
Conveyed	<Signal> Control
Information Source	<Property> : Vehicle Occupant
Information Target	<Property> : Climate Control Unit
Name	ItemFlow9
Namespace	<Package> 3 System Context
Qualified Name	VehicleCCU_Problem::1 Problem Domain::1 Black Box::3 System Context::ItemFlow9
Realization	
Realizing Activity Edge	
Realizing Connector	
Realizing Message	

Figure 68: Conveying the *Control* Signal



<Information Flow>

None

Stereotype display

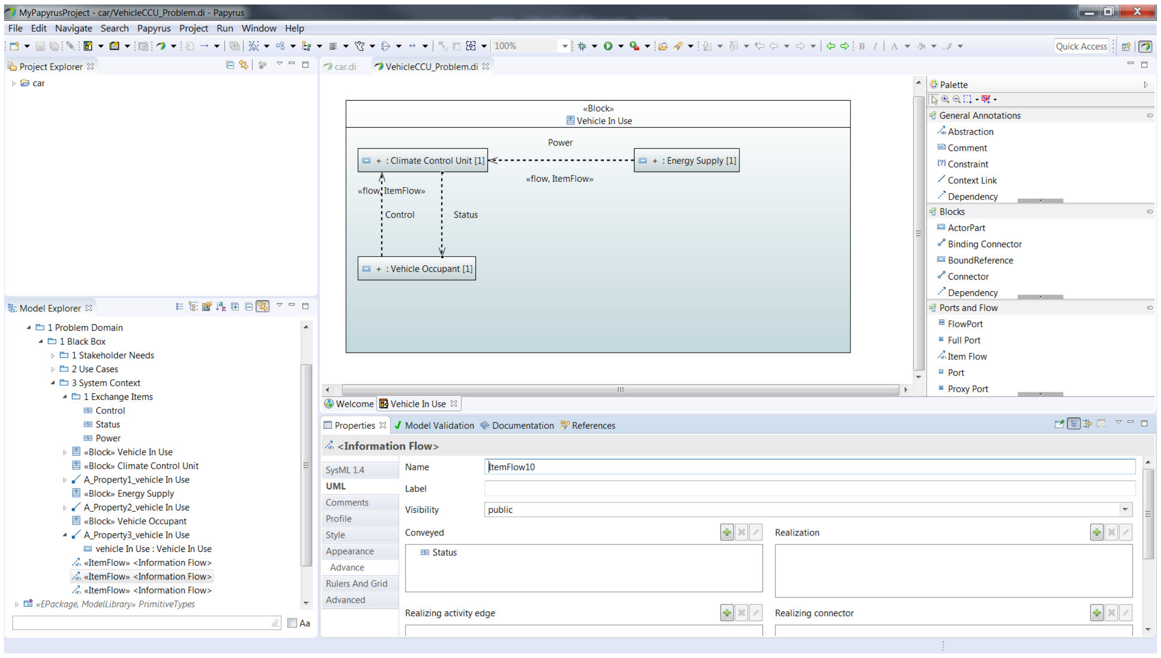
	Name Depth	Visible	In Braces	In Comment	In Compartment
⊟ <Information Flow>					
⊟ ItemFlow	none	☐	N/A	☐	N/A
itemProperty			N/A	☑	N/A

☐ Element icon

☐ Shadow

Figure 69: Item Flow Visibility

8. Follow the same procedures to create the flow for the *Status* signal From the *Climate Control Unit* To the *Vehicle Occupant* and the flow of *Power* From the *Energy Supply* To the *Climate Control Unit*. Your screen should look like (figure 70).



<Information Flow>

Name: ItemFlow10

Label:

Visibility: public

Conveyed: Status

Realization:

Realizing activity edge:

Realizing connector:

Figure 70: Item Flows

2.2.7 Step 7. Measures of Effectiveness

In this section we will refine the stakeholder needs with Measures of Effectiveness (MoE's) which add quantitative measurable characteristics of the system to serve as high-level performance indicators that would be monitored within the solution domain model. To do this we will create some value types to be used with our MoE's. Let's get started.

1. Create two new packages under the *1 Black Box* package; one named *4 Measures of Effectiveness* and another named *5 Types* (figure 71).

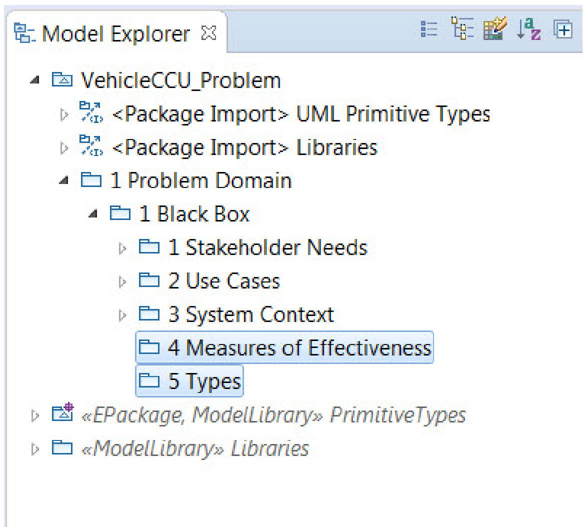


Figure 71: *Measures of Effectiveness* package and *Types* package.

2. Now we will work in the *5 Types* package to create our data types. Let's create a new package diagram to hold our data types. Right-click the *5 Types* package and select **New Diagram** and then select **SysML 1.4 Package Diagram** and name it *Types* (figure 72).

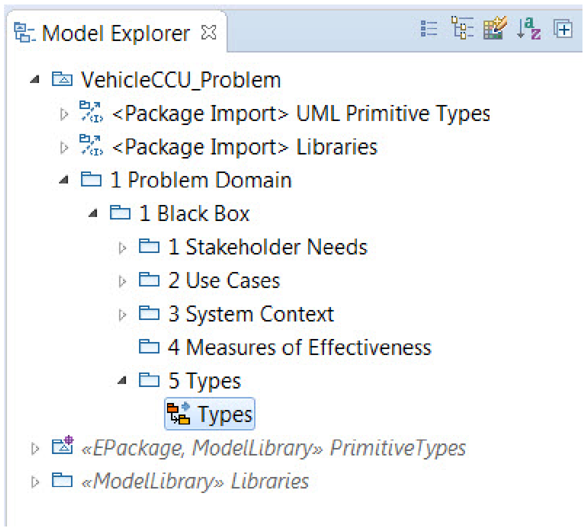


Figure 72: *Types* Package Diagram.

3. Now we will set up our MoE stereotype. Right-click on the *5 Types* package and select **New Child** and then **Stereotype** (figure 73) and name the stereotype *«moe»* (figure 74).

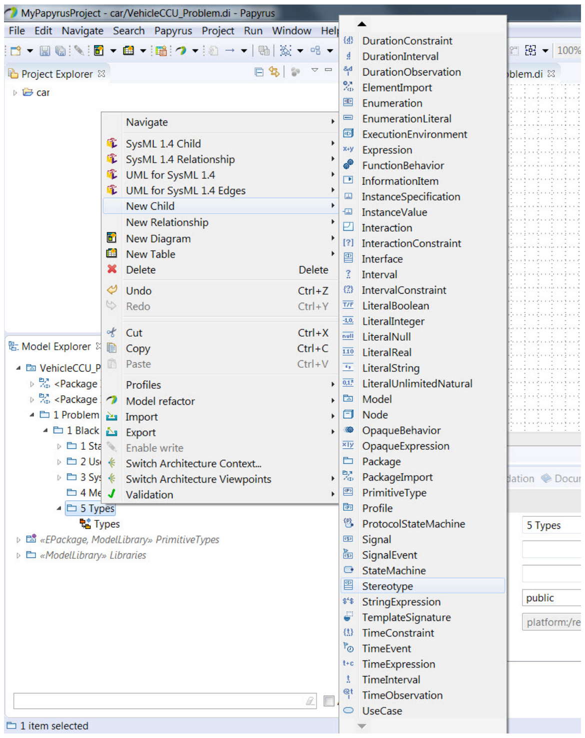


Figure 73: Creating the MoE stereotype.

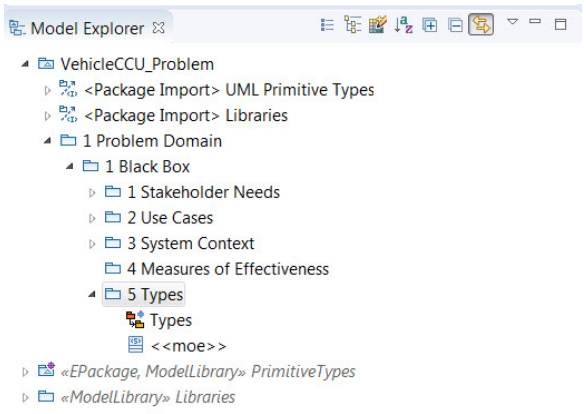


Figure 74: Stereotype Created.

4. Now we will create two value types to be used with our MOE's. Right-click the *«moe»* stereotype that you just created and select **SysML 1.4 Child** and then **ValueType** (figure 75). Name the value type *dBA*. This is our sound level value type measured in decibels. Create another value type and name this one *Mass[kilograms]*. Your *Model Explorer* window should look like (figure 76).

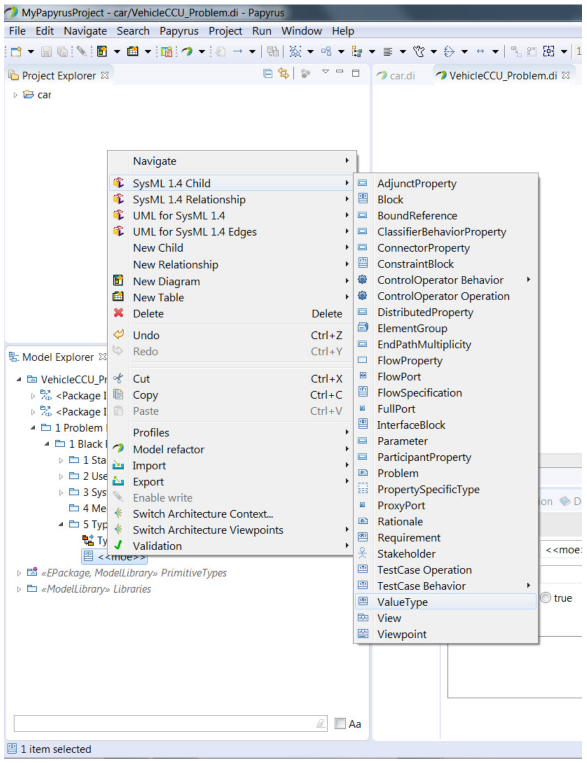
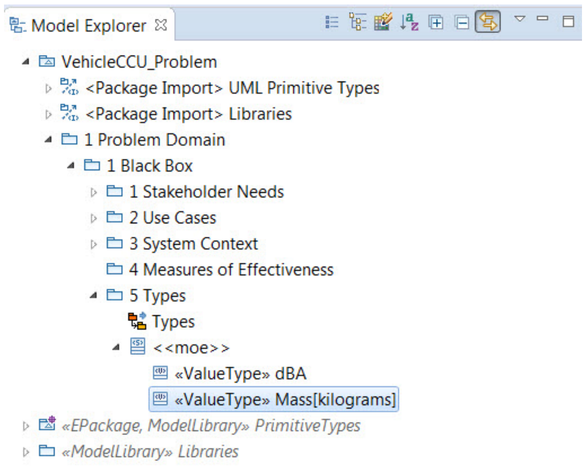


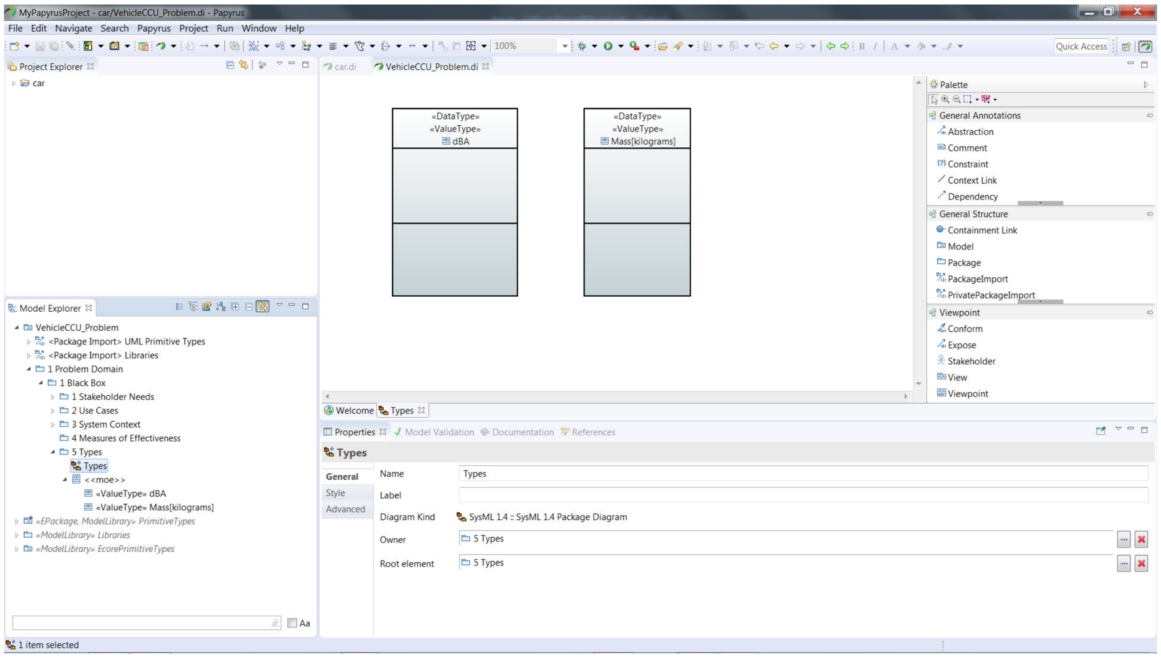
Figure 75: Value Type Menu Item.



```
VehicleCCU_Problem
  <Package Import> UML Primitive Types
  <Package Import> Libraries
  1 Problem Domain
    1 Black Box
      1 Stakeholder Needs
      2 Use Cases
      3 System Context
      4 Measures of Effectiveness
      5 Types
        Types
        <<moe>>
          «ValueType» dBA
          «ValueType» Mass[kilograms]
«EPackage, ModelLibrary» PrimitiveTypes
«ModelLibrary» Libraries
```

Figure 76: Value Types Created.

5. Now Click and Drag the *dBA* value type from the *Model Explorer* window to the *Types* package diagram in the main window. Do the same with the *Mass[kilograms]* value type. Your screen should now look like (figure 77).



```
VehicleCCU_Problem
  <Package Import> UML Primitive Types
  <Package Import> Libraries
  1 Problem Domain
    1 Black Box
      1 Stakeholder Needs
      2 Use Cases
      3 System Context
      4 Measures of Effectiveness
      5 Types
        Types
        <<moe>>
          «ValueType» dBA
          «ValueType» Mass[kilograms]
«EPackage, ModelLibrary» PrimitiveTypes
«ModelLibrary» Libraries
«ModelLibrary» EcorePrimitiveTypes
```

Types

General		
Style	Name	Types
Advanced	Label	
	Diagram Kind	SysML 1.4 :: SysML 1.4 Package Diagram
	Owner	5 Types
	Root element	5 Types

Figure 77: Value Types in Package Diagram.

6. Now that we have created our two value types let's get to our MOE's. Right-click the *4 Measures of Effectiveness* package in the *Model Explorer* and select **New Diagram** and then **SysML 1.4 Block Definition Diagram** (figure 78) and name it *Measures of Effectiveness*. Your *Model Explorer* window should look like (figure 79).

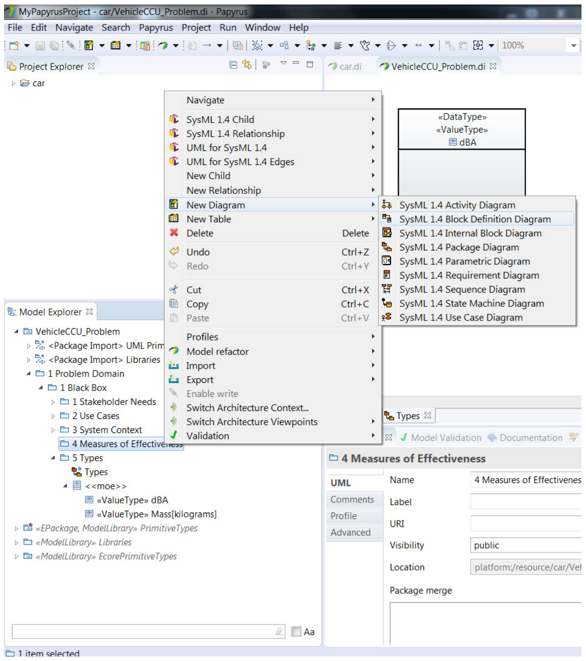

Navigate
SysML 1.4 Child
SysML 1.4 Relationship
UML for SysML 1.4
UML for SysML 1.4 Edges
New Child
New Relationship
New Diagram
New Table
Delete
Undo
Redo
Cut
Copy
Paste
Profiles
Model refactor
Import
Export
Enable write
Switch Architecture Context...
Switch Architecture Viewpoints
Validation
SysML 1.4 Activity Diagram
SysML 1.4 Block Definition Diagram
SysML 1.4 Internal Block Diagram
SysML 1.4 Package Diagram
SysML 1.4 Parametric Diagram
SysML 1.4 Requirement Diagram
SysML 1.4 Sequence Diagram
SysML 1.4 State Machine Diagram
SysML 1.4 Use Case Diagram
«DataType» «ValueType» dBA
Model Explorer
VehicleCCU_Problem
1 Problem Domain
1 Black Box
4 Measures of Effectiveness
5 Types
«ValueType» dBA
«ValueType» Mass[kilograms]
4 Measures of Effectiveness
Name
Visibility
public

Figure 78: Create Block Definition Diagram.

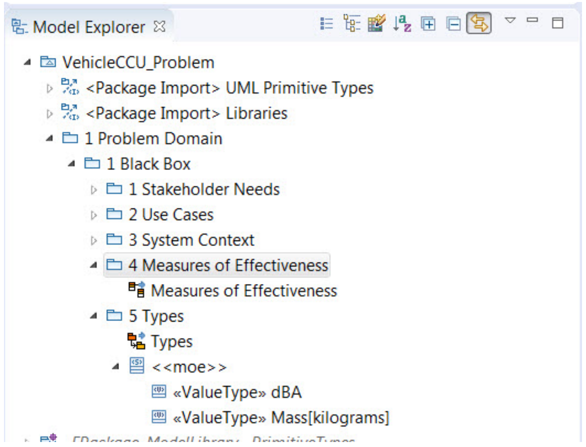
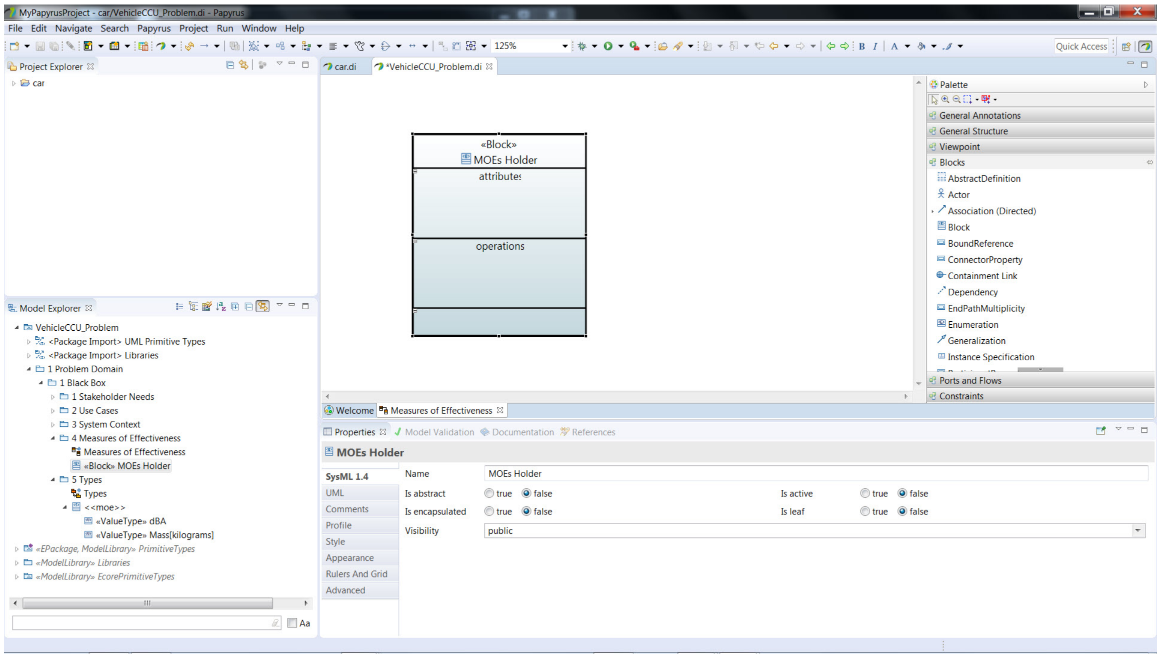

Model Explorer
VehicleCCU_Problem
<Package Import> UML Primitive Types
<Package Import> Libraries
1 Problem Domain
1 Black Box
1 Stakeholder Needs
2 Use Cases
3 System Context
4 Measures of Effectiveness
Measures of Effectiveness
5 Types
Types
<<moe>>
«ValueType» dBA
«ValueType» Mass[kilograms]

Figure 79: *Measures of Effectiveness* BDD.

7. Now we will create a block to hold our MOE's. Using the **Blocks** tool palette on the right, click on **Block** and then click anywhere inside the *Measures of Effectiveness* block definition diagram in the main window to create the new block. Name this block *MOEs Holder* (figure 80).

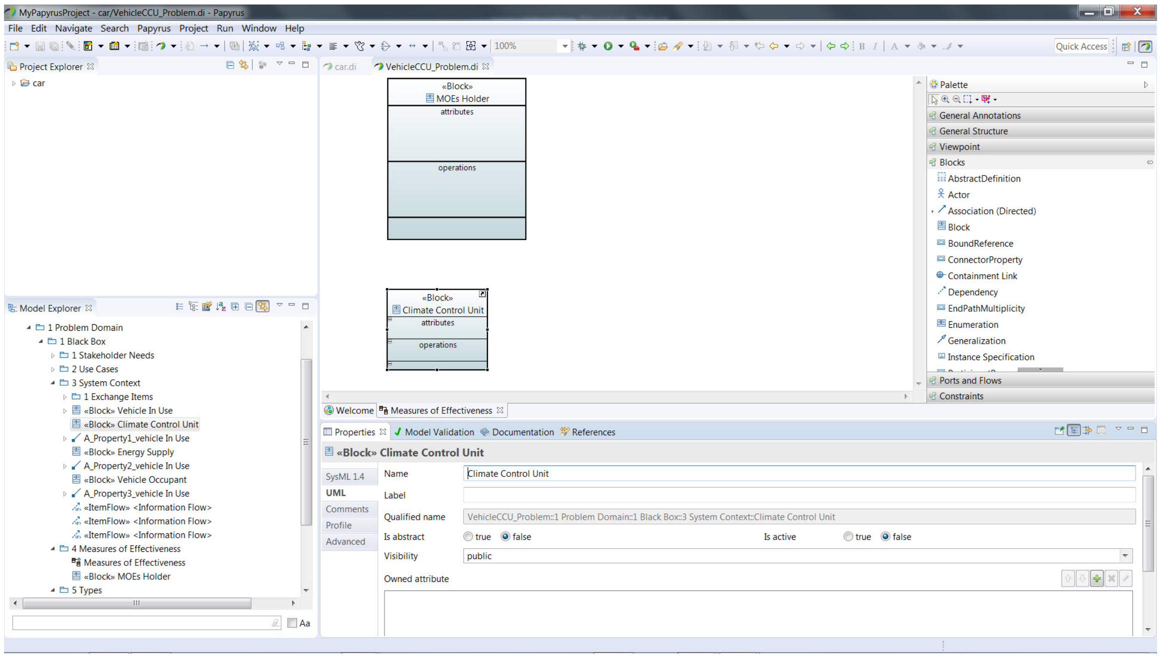


MOEs Holder

Name: MOEs Holder
Is abstract: ○ true [✓ ● false]
Is active: ○ true [✓ ● false]
Is encapsulated: ○ true [✓ ● false]
Is leaf: ○ true [✓ ● false]
Visibility: public

Figure 80: *MOEs Holder* Block.

8. Now we need to make this *MOEs Holder* block a supertype of the *Climate Control Unit* block. In the *Model Explorer* window expand the *3 Systems Context* package. Click and Drag the *Climate Control Unit* block to the *Measures of Effectiveness* block definition diagram (bdd) in the main window and position it underneath the *MOEs Holder* block (figure 81).

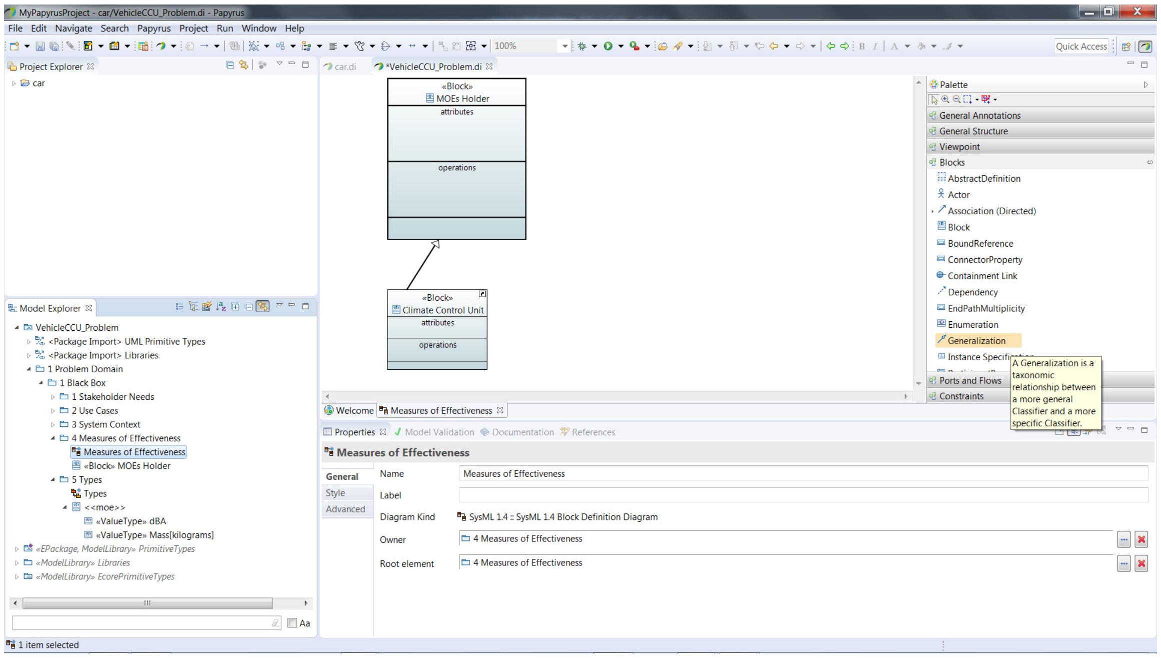


«Block» Climate Control Unit

Name: Climate Control Unit
Label:
Qualified name: VehicleCCU_Problem::1 Problem Domain::1 Black Box::3 System Context::Climate Control Unit
Is abstract: ○ true [✓ ● false]
Is active: ○ true [✓ ● false]
Visibility: public
Owned attribute

Figure 81: *Climate Control Unit* block in *Measures of Effectiveness* bdd.

9. Using the **Blocks** tool palette on the right, select **Generalization** and then in the block definition diagram Click on the *Climate Control Unit* block and then Click on the *MOEs Holder* block. The generalization arrow will appear on the diagram (figure 82).



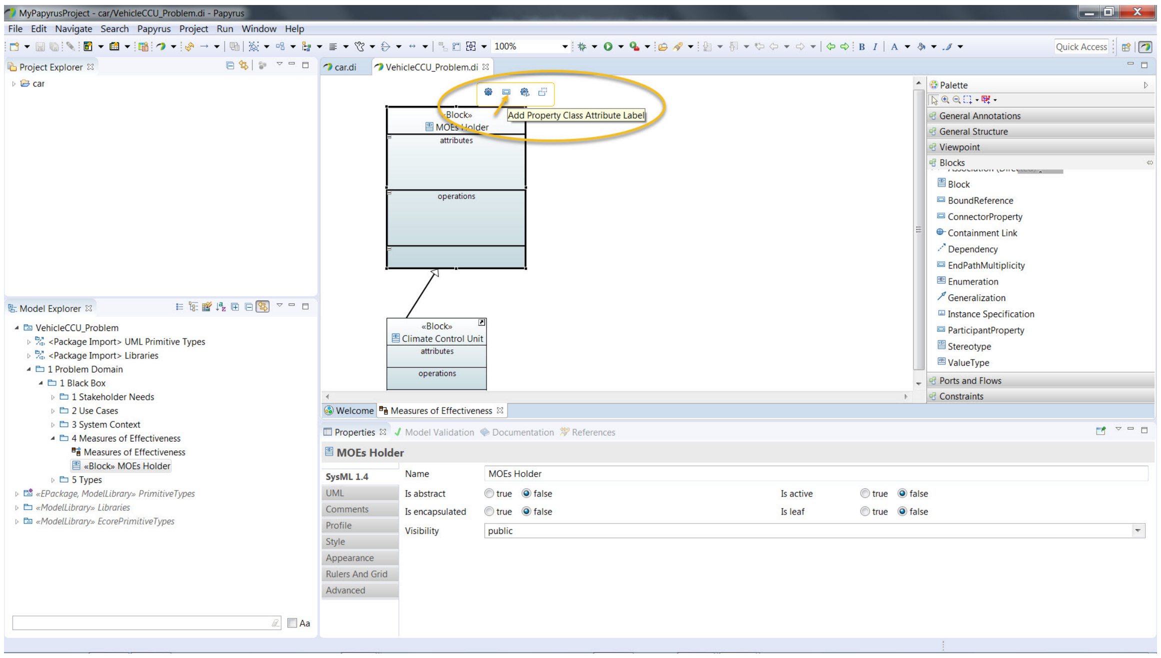
A Generalization is a taxonomic relationship between a more general Classifier and a more specific Classifier.

Measures of Effectiveness

Name: Measures of Effectiveness
Label:
Diagram Kind: SysML 1.4 :: SysML 1.4 Block Definition Diagram
Owner: 4 Measures of Effectiveness
Root element: 4 Measures of Effectiveness

Figure 82: Creating Generalization in the bdd.

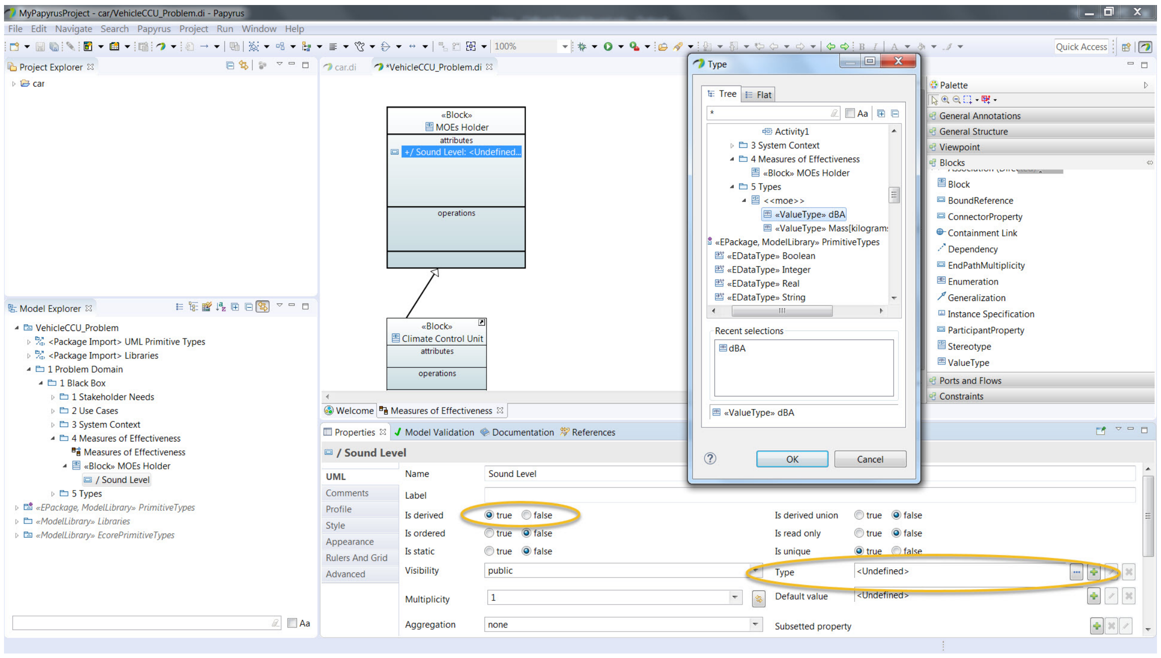
10. Now we will define our MOE attributes. Click on the *MOEs Holder* block and you should see s pop-up menu at the top of the block. Select **Add Property Class Attribute Label** (figure 83) and hit <Enter>. Go to the **Properties** window at the bottom of the screen and select the **UML** tab. In the **Name** field type *Sound Level*. Select **true** for the **Is derived** attribute. This means that the sound level will be a calculated value. Finally, in the **Type** field click on the three dots to the right of the field to open the pop-up menu to define the type. Select the *dBA* value type that we created in the *Types* package and then click <OK> (figure 84) .



MOEs Holder

Name	MOEs Holder		
Is abstract	true [✓ false]	Is active	true [✓ false]
Is encapsulated	true [✓ false]	Is leaf	true [✓ false]
Visibility	public		

Figure 83: Class Attribute Pop-Up Menu.

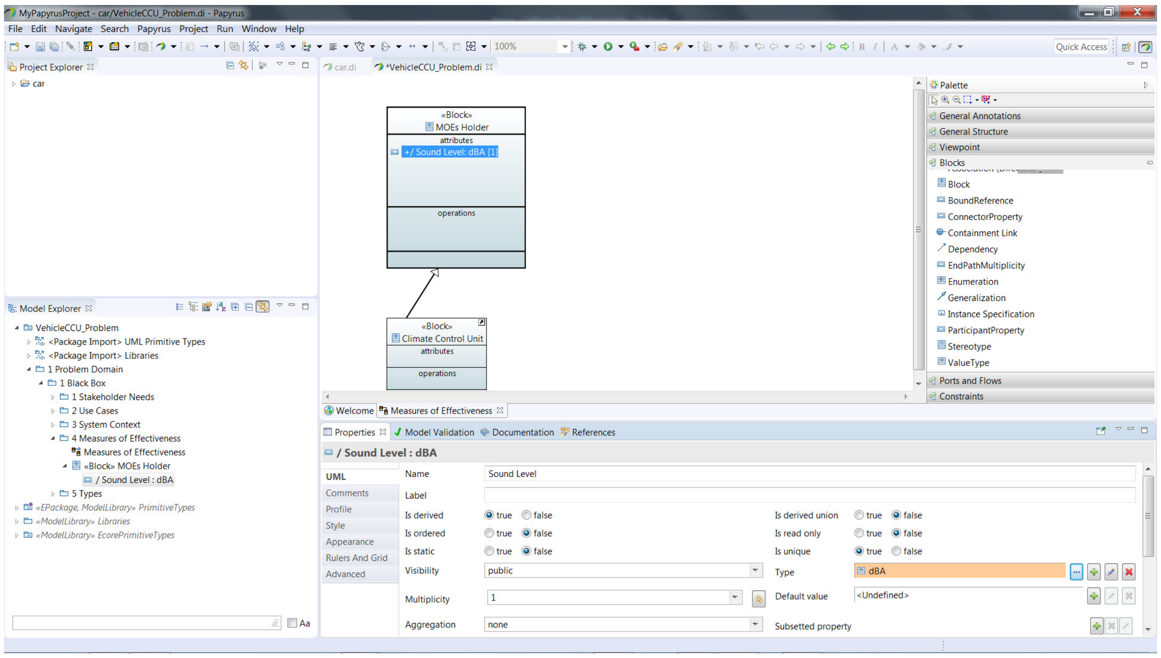


/ Sound Level

Name	Sound Level		
Label			
Is derived	[✓ true] false	Is derived union	true [✓ false]
Is ordered	true [✓ false]	Is read only	true [✓ false]
Is static	true [✓ false]	Is unique	[✓ true] false
Visibility	public	Type	<Undefined>
Multiplicity	1	Default value	<Undefined>
Aggregation	none	Subsetted property	

Figure 84: *Sound Level* Attribute Properties.

11. Your diagram should look like (figure 85).



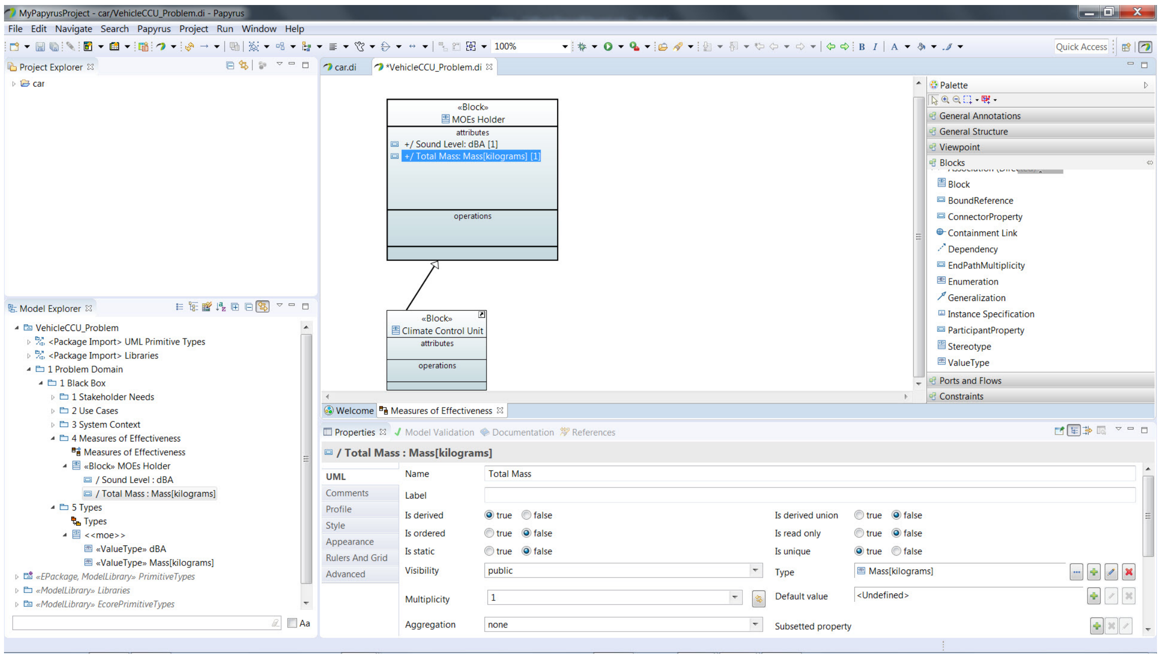
Model Explorer
- VehicleCCU_Problem
  - <Package Import> UML Primitive Types
  - <Package Import> Libraries
  - 1 Problem Domain
    - 1 Black Box
      - 1 Stakeholder Needs
      - 2 Use Cases
      - 3 System Context
      - 4 Measures of Effectiveness
        - Measures of Effectiveness
        - «Block» MOEs Holder
          - / Sound Level : dBA
      - 5 Types
- «EPackage, ModelLibrary» PrimitiveTypes
- «ModelLibrary» Libraries
- «ModelLibrary» EcorePrimitiveTypes

/ Sound Level : dBA

Name	Sound Level		
Label			
Is derived	[✓ true] false	Is derived union	true [✓ false]
Is ordered	true [✓ false]	Is read only	true [✓ false]
Is static	true [✓ false]	Is unique	[✓ true] false
Visibility	public	Type	dBA
Multiplicity	1	Default value	<Undefined>
Aggregation	none	Subsetted property	

Figure 85: *Sound Level* Attribute Created.

12. Follow that same procedures to create another attribute in the *MOEs Holder* block named *Total Mass*. For its **Type** property select the *Mass* value type that we created earlier in the *5 Types* package. Your diagram should look like (figure 86). We have now finished our Black Box analysis and will proceed to our White Box analysis.



Model Explorer
- VehicleCCU_Problem
  - <Package Import> UML Primitive Types
  - <Package Import> Libraries
  - 1 Problem Domain
    - 1 Black Box
      - 1 Stakeholder Needs
      - 2 Use Cases
      - 3 System Context
      - 4 Measures of Effectiveness
        - Measures of Effectiveness
        - «Block» MOEs Holder
          - / Sound Level : dBA
          - / Total Mass : Mass[kilograms]
      - 5 Types
        - Types
        - <<moe>>
          - «ValueType» dBA
          - «ValueType» Mass[kilograms]
- «EPackage, ModelLibrary» PrimitiveTypes
- «ModelLibrary» Libraries
- «ModelLibrary» EcorePrimitiveTypes

/ Total Mass : Mass[kilograms]

Name	Total Mass		
Label			
Is derived	[✓ true] false	Is derived union	true [✓ false]
Is ordered	true [✓ false]	Is read only	true [✓ false]
Is static	true [✓ false]	Is unique	[✓ true] false
Visibility	public	Type	Mass[kilograms]
Multiplicity	1	Default value	<Undefined>
Aggregation	none	Subsetted property	

Figure 86: *Total Mass* Attribute Created.