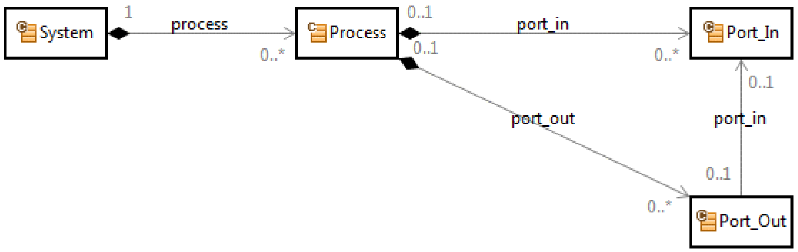
Tutorial

Domain Diagram Type II

Tutorial	Actifsource Tutorial – Simple Service
Required Time	• 70 Minutes
Prerequisites	• Actifsource Tutorial – Installing Actifsource • Actifsource Tutorial – Simple Service • Actifsource Tutorial – Domain Diagram Type
Goal	• Define Diagram Types to create and edit domain models in a graphical editor
Topics covered	• Create a Diagram Type • Working with Diagram Editor • Define shapes, figures and ports to use in Diagram Editor • Add conditions to figures • Add a search function to a domain diagram • Add notes to domain diagrams • Add labels to ports • Insert links in Domain Diagrams to (external) diagrams (e.g. UML State Machines)
Notation	↳ To do ⓘ Information • **Bold**: Terms from actifsource or other technologies and tools • <u>**Bold underlined**</u>: actifsource Resources • <u>Underlined</u>: User Resources • <u>*UnderlinedItalics*</u>: Resource Functions • `Monospaced`: User input • *Italics*: Important terms in current situation
Disclaimer	The authors do not accept any liability arising out of the application or use of any information or equipment described herein. The information contained within this document is by its very nature incomplete. Therefore the authors accept no responsibility for the precise accuracy of the documentation contained herein. It should be used rather as a guide and starting point.
Contact	**actifsource GmbH** Täfernstrasse 37 5405 Baden-Dättwil Switzerland <u>www.actifsource.com</u>
Trademark	**actifsource** is a registered trademark of **actifsource GmbH** in Switzerland, the EU, USA, and China. Other names appearing on the site may be trademarks of their respective owners.
Compatibility	Created with **actifsource** Version 6.8.1

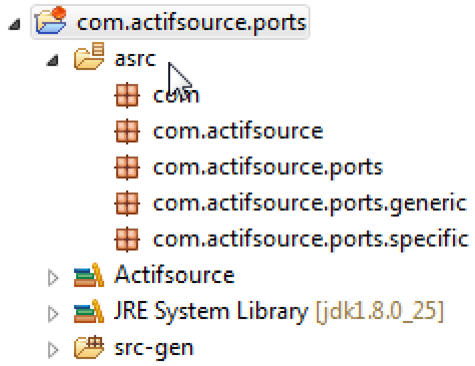
- Create a new (Domain) Diagram Type for a simple meta model of systems composed of processes that have (outgoing) out-ports and (incoming) in-ports to communicate with each other:



- Define the properties of the Domain Diagram such that names of processes can be edited, ports can be created and out-ports (Port_Out) can be connected to in-ports (Port_In).
- Add a search function that allows you to search for specific processes in a Domain Diagram
- Add links to (existing) state diagrams to Domain Diagrams, e.g., to associate an UML state machines with a specific process to describe its behavior

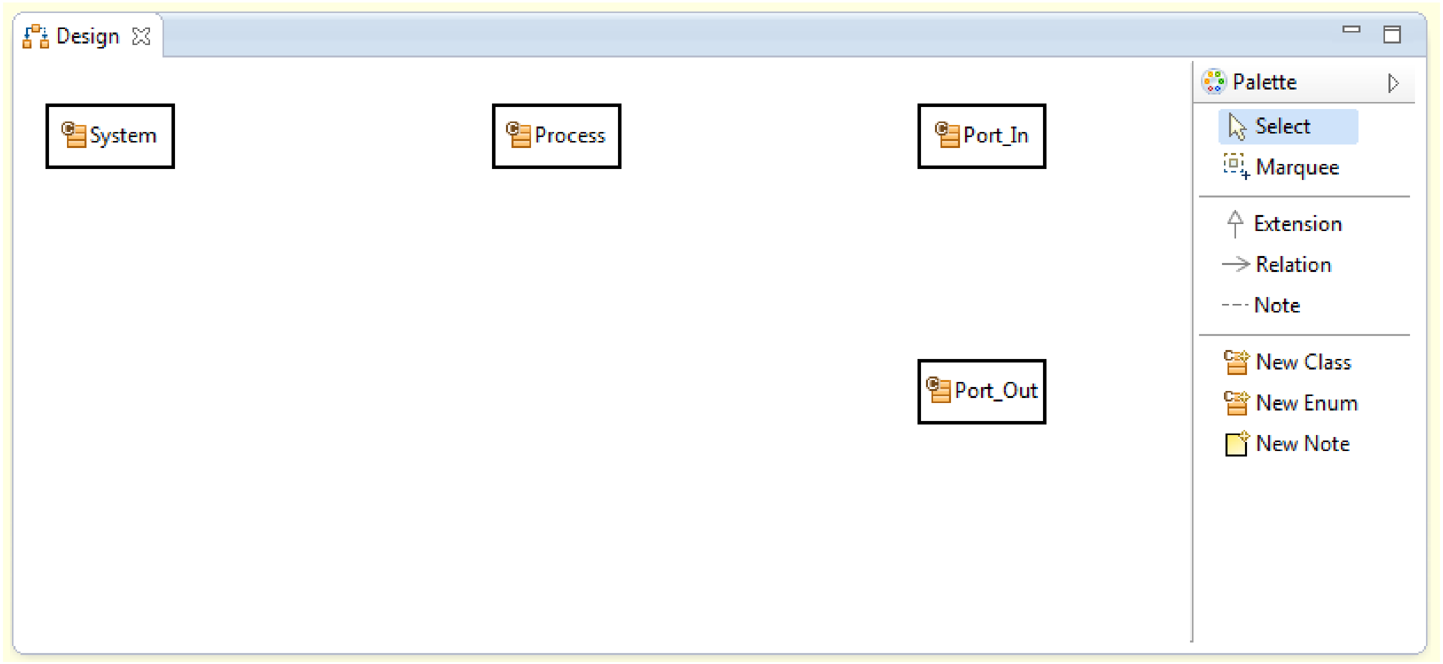
Part I: Preparation

- Prepare a new **actifsource Project** named com.actifsource.ports as seen in the *Actifsource Tutorial – Simple Service*
- Use the following package structure

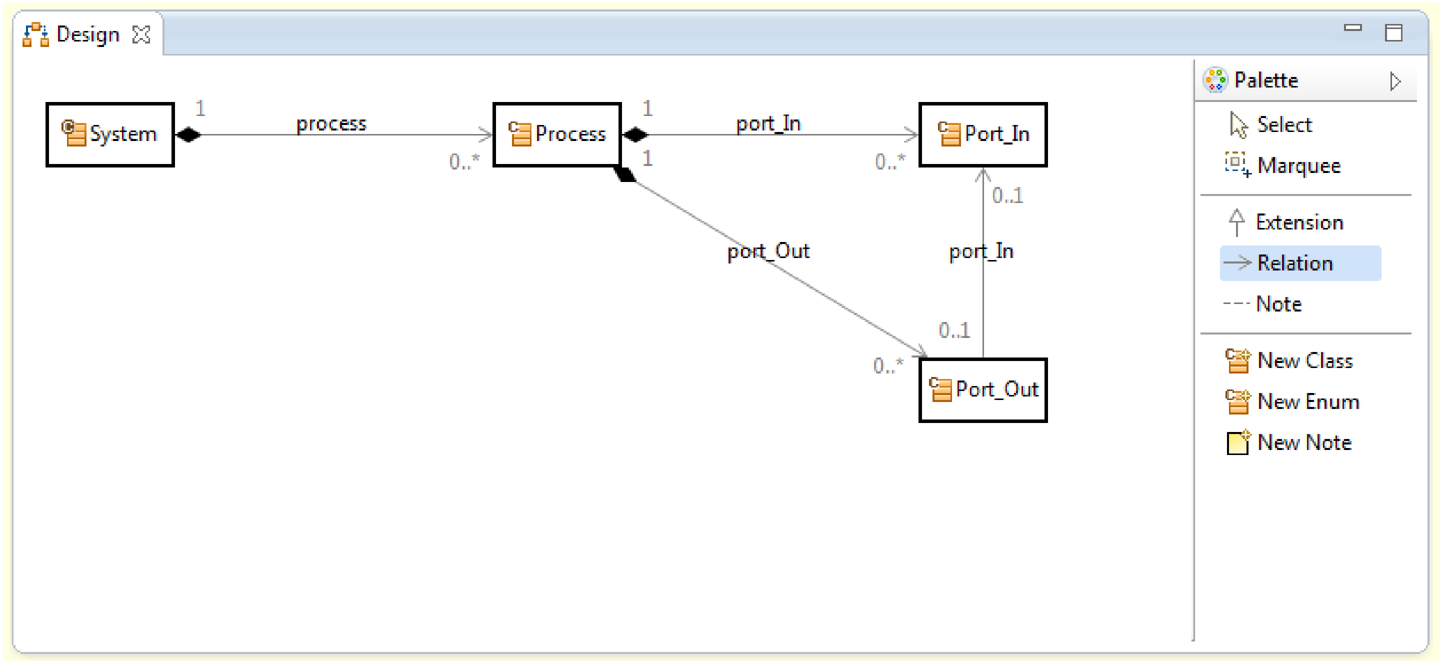


```
com.actifsource.ports
  asrc
    com
    com.actifsource
    com.actifsource.ports
    com.actifsource.ports.generic
    com.actifsource.ports.specific
  Actifsource
  JRE System Library [jdk1.8.0_25]
  src-gen
```

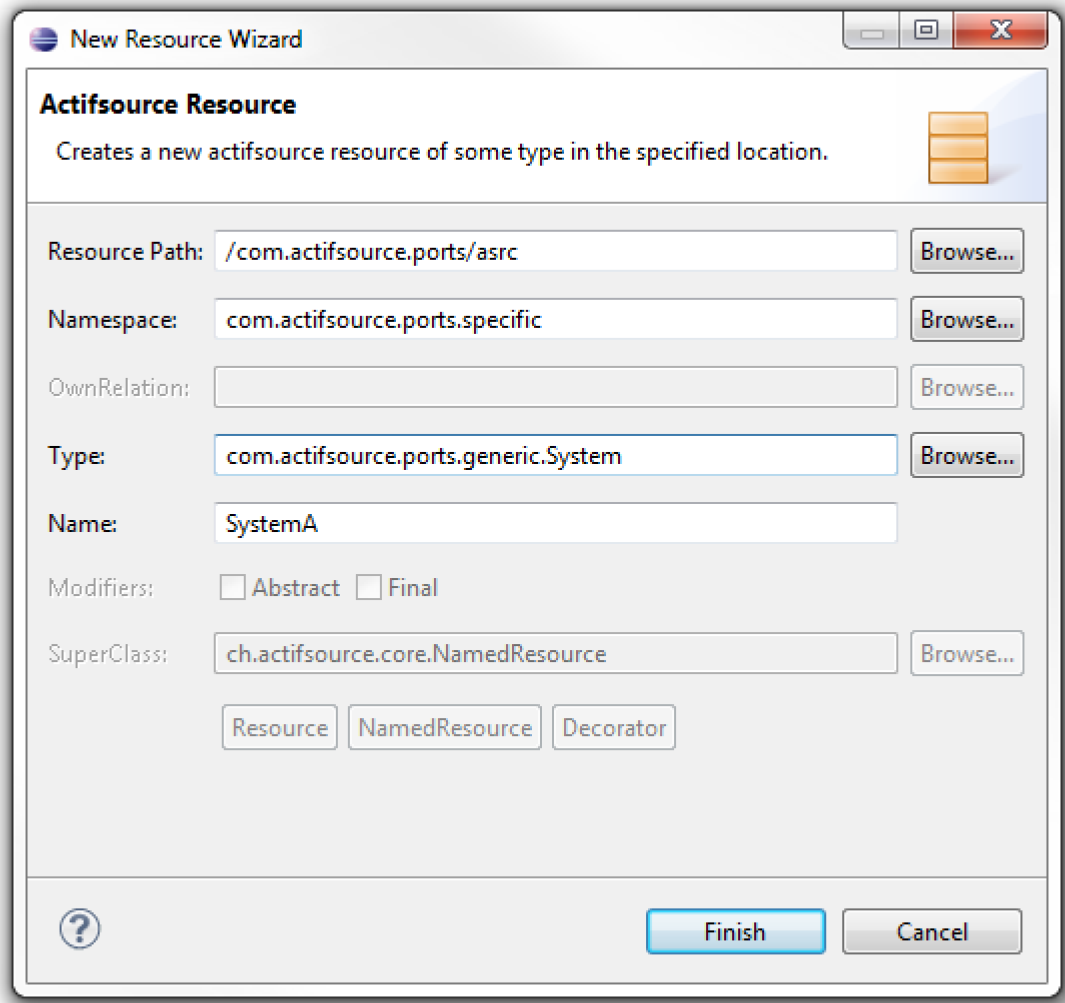
Preparation



- Create a **ClassDiagram** named *Design* in the **Package** *generic* using the **DiagramEditor**
- Create the following **Classes:**
  - System, Process**,** Port_In, Port_Out



- Insert an **OwnRelation** between
  - System and Process
  - Process and Port_In
  - Process and Port_Out
- Insert a **UseRelation** between
  - Port_Out and Port_In
- Adjust the **Cardinalities** as shown above



Resource Path: /com.actifsource.ports/asrc

Namespace: com.actifsource.ports.specific

OwnRelation:

Type: com.actifsource.ports.generic.System

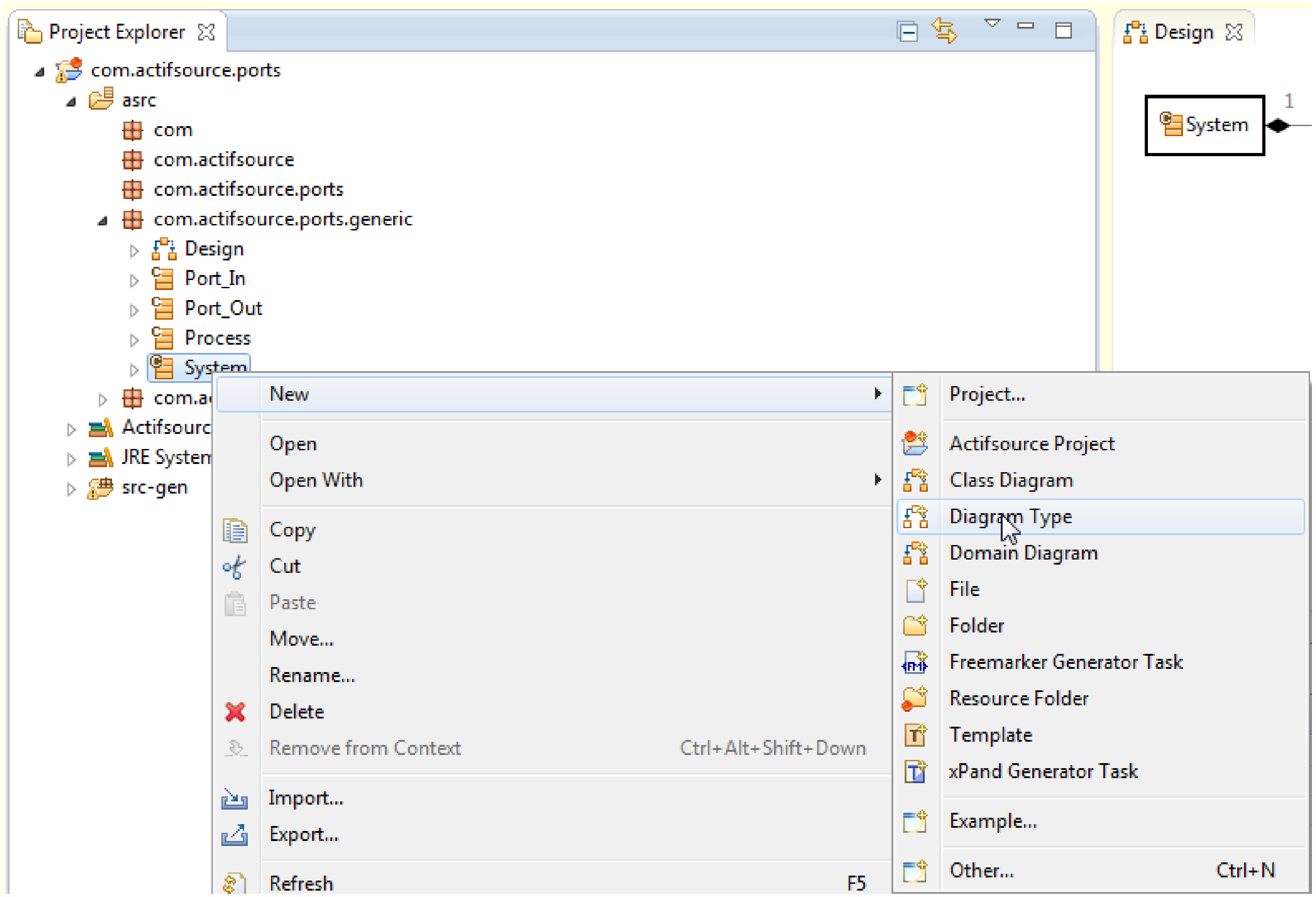
Name: SystemA

Modifiers: ☐ Abstract ☐ Final

SuperClass: ch.actifsource.core.NamedResource

- Create a new **<u>Resource</u>** of type <u>System</u> in the package com.actifsource.ports.specific (Right-click on the package and choose **New->Resource** from the menu)
- Give the name `SystemA` to the the newly created resource in the **New Resource Wizard**

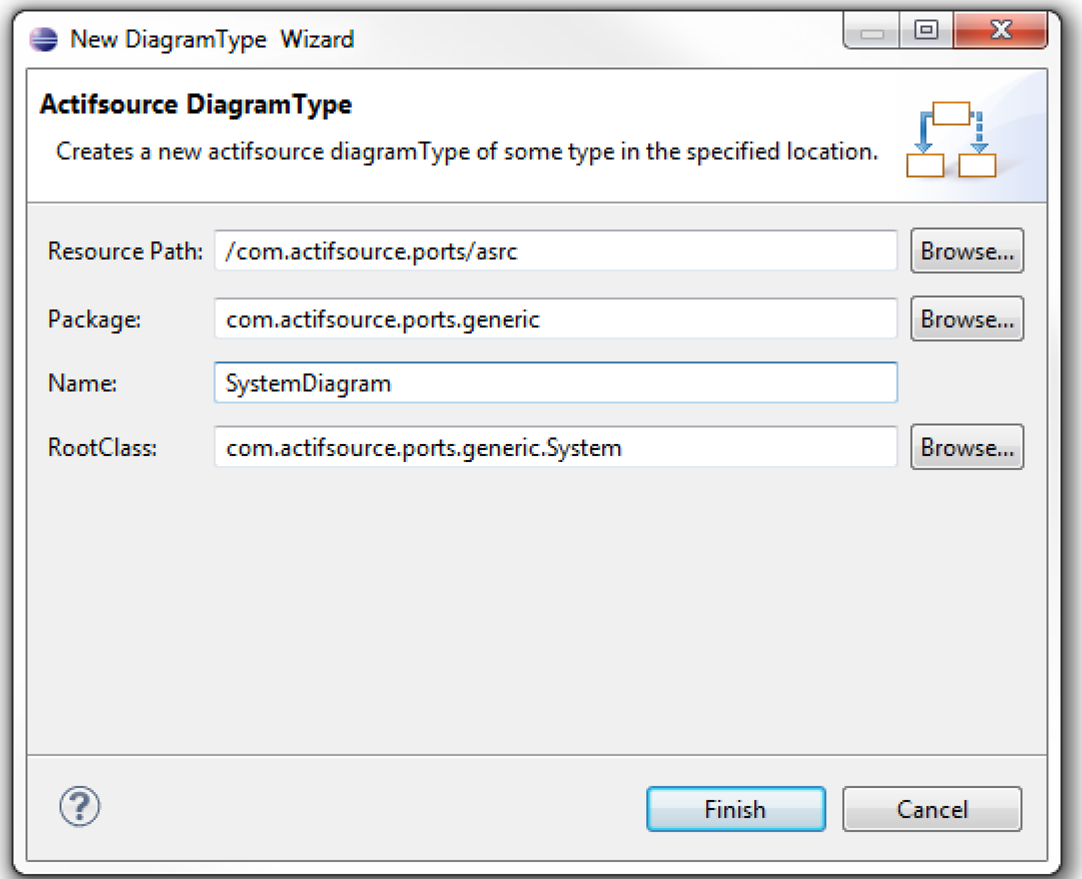
Part II: Define a new Diagram Type



We create a new Diagram Type called System in order to define properties of **Domain Diagrams** of Systems:

↳ Select the resource System in the package com.actifsource.ports.generic and choose **New -> Diagram Type** from the menu

Define a new Diagram Type

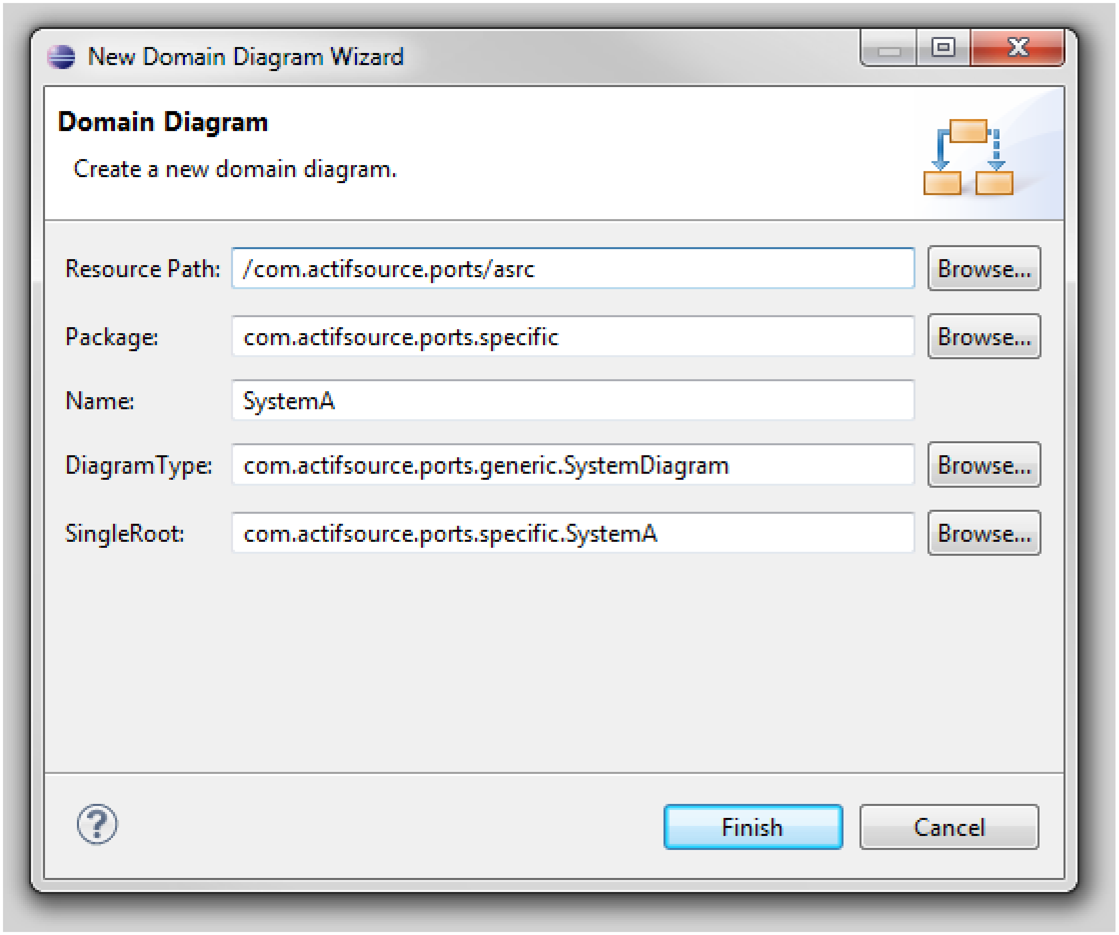

New DiagramType Wizard

Actifsource DiagramType

Creates a new actifsource diagramType of some type in the specified location.

Resource Path:	/com.actifsource.ports/asrc	Browse...
Package:	com.actifsource.ports.generic	Browse...
Name:	SystemDiagram	
RootClass:	com.actifsource.ports.generic.System	Browse...

- Enter `SystemDiagram` as name for the newly created DiagramType in the **New DiagramType Wizard**
- Make sure that com.actifsource.ports.generic.Process has (automatically) been chosen as **RootClass**



Domain Diagram

Create a new domain diagram.

Resource Path: /com.actifsource.ports/asrc

Package: com.actifsource.ports.specific

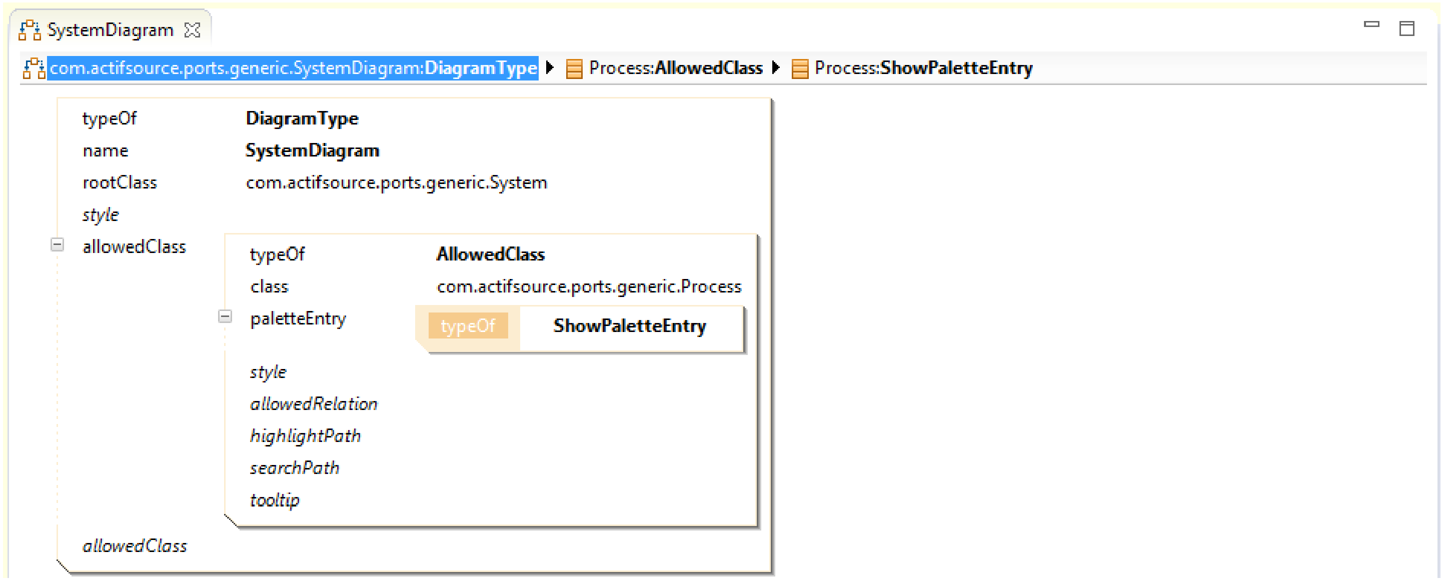
Name: SystemA

DiagramType: com.actifsource.ports.generic.SystemDiagram

SingleRoot: com.actifsource.ports.specific.SystemA

We create a new **Domain Diagram** for the resource SystemA:

- Select the resource SystemA and choose **New -> Domain Diagram** from the menu
- Enter *SystemA* as the name of the new diagram in the **New Domain Diagram Wizard**
- Click Finish


SystemDiagram

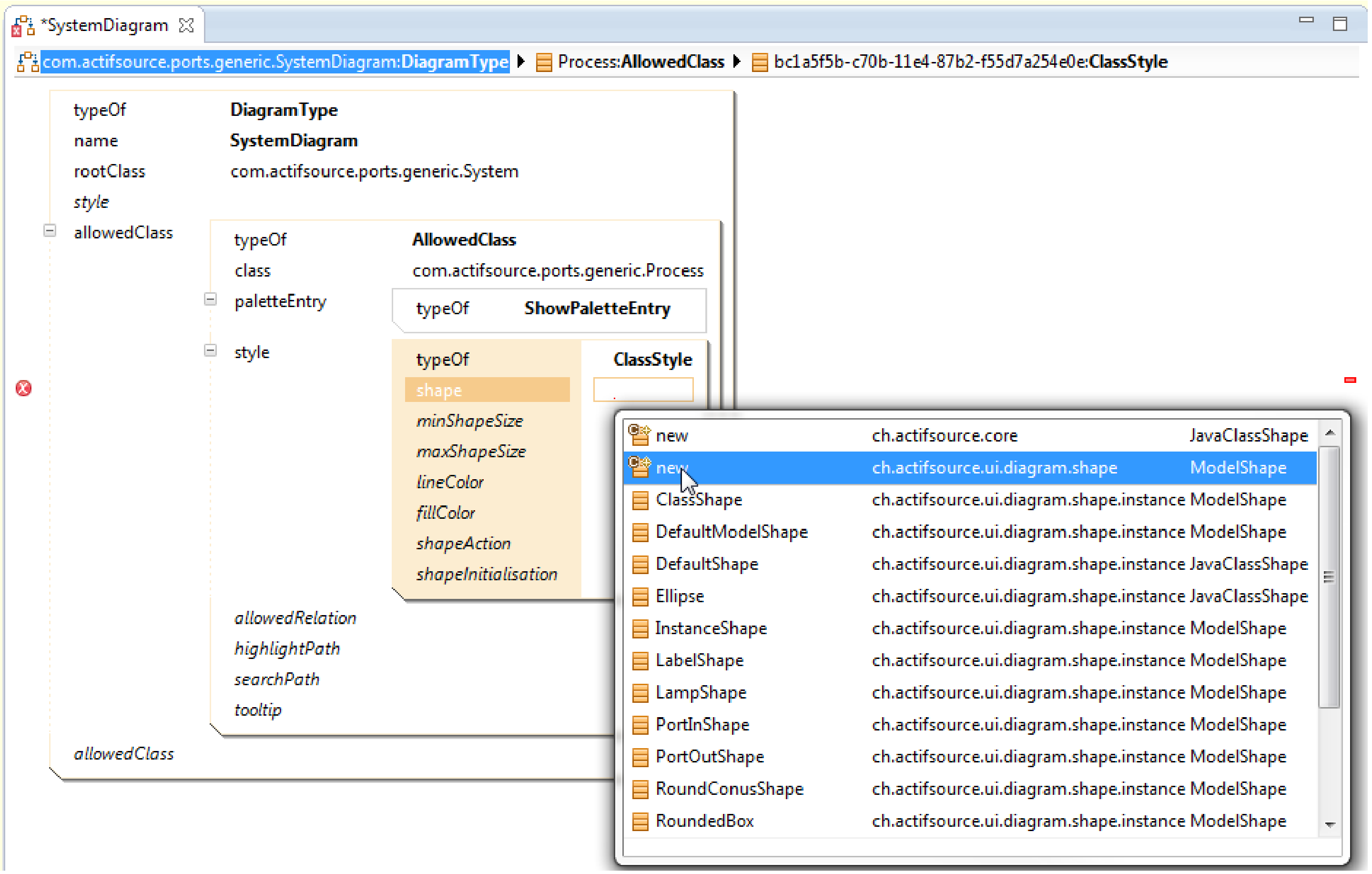
com.actifsource.ports.generic.SystemDiagram:DiagramType ▶ Process:AllowedClass ▶ Process:ShowPaletteEntry

typeOf	DiagramType		
name	SystemDiagram		
rootClass	com.actifsource.ports.generic.System		
style			
allowedClass	typeOf	AllowedClass	
	class	com.actifsource.ports.generic.Process	
	paletteEntry	typeOf	ShowPaletteEntry
	style		
	allowedRelation		
	highlightPath		
	searchPath		
	tooltip		
allowedClass			

Next, we create a shape and a figure for the class Process in order to define how elements of type Process are displayed and handled in the **Diagram Editor**:

- Open SystemDiagram in the **Resource Editor**
- Create a statement **allowedClass** refering to an **AllowedClass** with class com.actifsource.ports.generic.Process
- As **paletteEntry** choose the type **ShowPaletteEntry**

Define a new Diagram Type


*SystemDiagram

com.actifsource.ports.generic.SystemDiagram:DiagramType ▸ Process:AllowedClass ▸ bc1a5f5b-c70b-11e4-87b2-f55d7a254e0e:ClassStyle

typeOf	**DiagramType**
name	**SystemDiagram**
rootClass	com.actifsource.ports.generic.System
style	
allowedClass	

typeOf	**AllowedClass**
class	com.actifsource.ports.generic.Process
paletteEntry	typeOf **ShowPaletteEntry**
style	

typeOf	**ClassStyle**
shape	
minShapeSize	
maxShapeSize	
lineColor	
fillColor	
shapeAction	
shapeInitialisation	

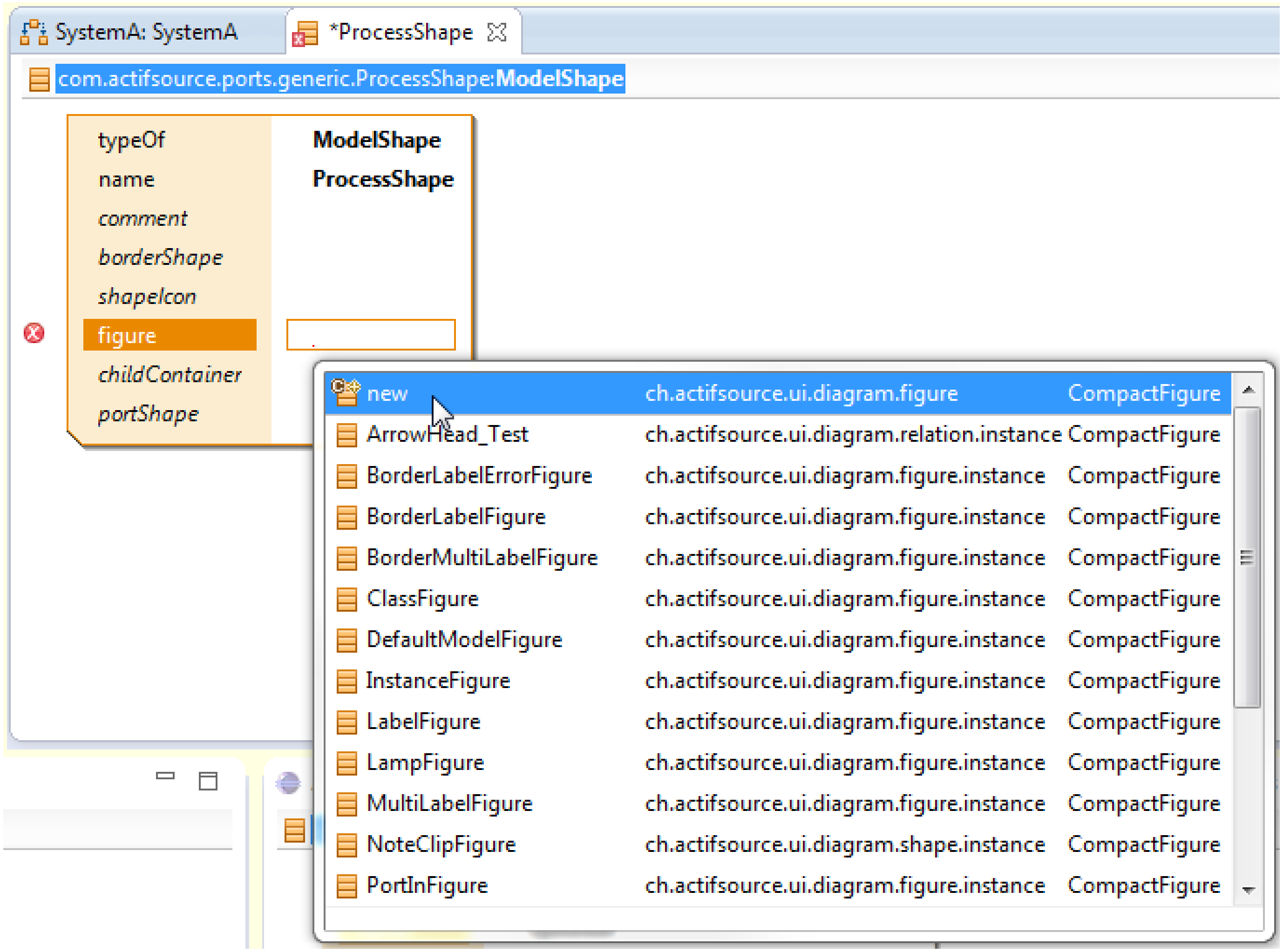
allowedRelation
highlightPath
searchPath
tooltip

allowedClass

new	ch.actifsource.core	JavaClassShape
new	ch.actifsource.ui.diagram.shape	ModelShape
ClassShape	ch.actifsource.ui.diagram.shape.instance	ModelShape
DefaultModelShape	ch.actifsource.ui.diagram.shape.instance	ModelShape
DefaultShape	ch.actifsource.ui.diagram.shape.instance	JavaClassShape
Ellipse	ch.actifsource.ui.diagram.shape.instance	JavaClassShape
InstanceShape	ch.actifsource.ui.diagram.shape.instance	ModelShape
LabelShape	ch.actifsource.ui.diagram.shape.instance	ModelShape
LampShape	ch.actifsource.ui.diagram.shape.instance	ModelShape
PortInShape	ch.actifsource.ui.diagram.shape.instance	ModelShape
PortOutShape	ch.actifsource.ui.diagram.shape.instance	ModelShape
RoundConusShape	ch.actifsource.ui.diagram.shape.instance	ModelShape
RoundedBox	ch.actifsource.ui.diagram.shape.instance	ModelShape

↳ Create a new **ClassStyle** as **style** and add a new **ModelShape** as **shape** to it

Define a new Diagram Type

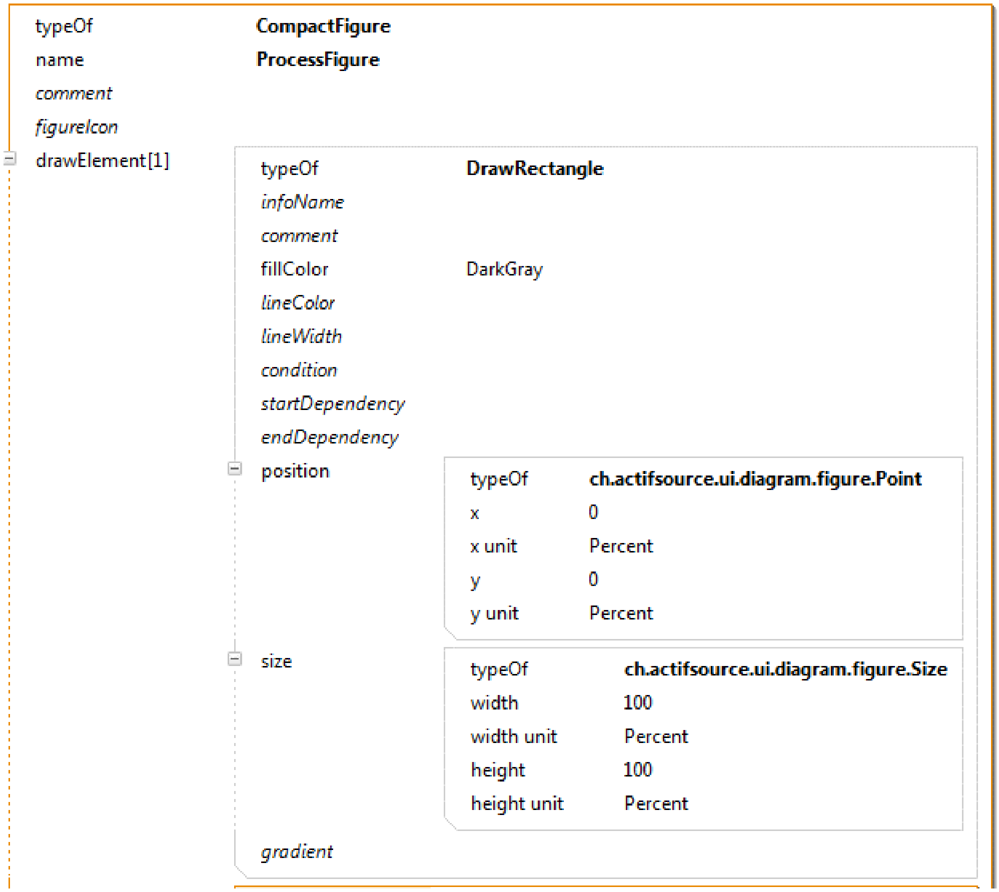


com.actifsource.ports.generic.ProcessShape:ModelShape

typeOf	ModelShape
name	ProcessShape
comment	
borderShape	
shapeIcon	
figure	
childContainer	
portShape	

new	ch.actifsource.ui.diagram.figure	CompactFigure
ArrowHead_Test	ch.actifsource.ui.diagram.relation.instance	CompactFigure
BorderLabelErrorFigure	ch.actifsource.ui.diagram.figure.instance	CompactFigure
BorderLabelFigure	ch.actifsource.ui.diagram.figure.instance	CompactFigure
BorderMultiLabelFigure	ch.actifsource.ui.diagram.figure.instance	CompactFigure
ClassFigure	ch.actifsource.ui.diagram.figure.instance	CompactFigure
DefaultModelFigure	ch.actifsource.ui.diagram.figure.instance	CompactFigure
InstanceFigure	ch.actifsource.ui.diagram.figure.instance	CompactFigure
LabelFigure	ch.actifsource.ui.diagram.figure.instance	CompactFigure
LampFigure	ch.actifsource.ui.diagram.figure.instance	CompactFigure
MultiLabelFigure	ch.actifsource.ui.diagram.figure.instance	CompactFigure
NoteClipFigure	ch.actifsource.ui.diagram.shape.instance	CompactFigure
PortInFigure	ch.actifsource.ui.diagram.figure.instance	CompactFigure

- Choose ProcessShape as name of the new **ModelShape**
- Create a statement **figure** that refers to a new **ch.actifsource.ui.diagram.figure.CompactFigure**



typeOf	CompactFigure
name	ProcessFigure
comment	
figureIcon	
drawElement[1]	

drawElement[1]:

typeOf	DrawRectangle
infoName	
comment	
fillColor	DarkGray
lineColor	
lineWidth	
condition	
startDependency	
endDependency	
position	
size	
gradient	

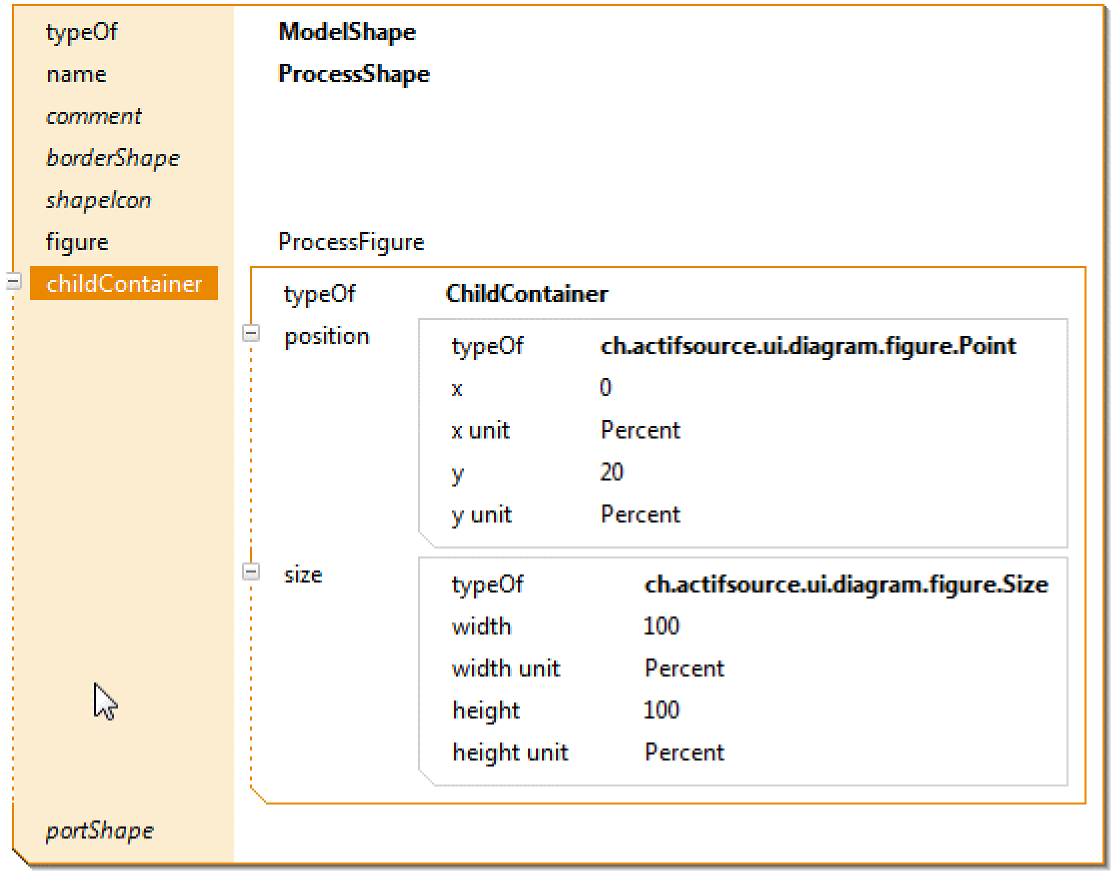
position:

typeOf	ch.actifsource.ui.diagram.figure.Point
x	0
x unit	Percent
y	0
y unit	Percent

size:

typeOf	ch.actifsource.ui.diagram.figure.Size
width	100
width unit	Percent
height	100
height unit	Percent

- Choose `ProcessFigure` as name of the new **CompactFigure**
- Create a new **drawElement** and choose the type **DrawRectangle** from the **Type Selection** dialog
- Define a **fillColor** by choosing **DarkGray** with the help of the Content Assist
- Create a **position** (x=0, y=0) and a **size** (width = 100%, height=100%) statement as shown above
- ⓘ Note that the conventions for drawing graphic elements follows in general the conventions used in Java native libraries (e.g. Java AWT). This means that (x=0,y=0) is positioned in the upper left hand corner. The grid is then numbered in a positive direction on the *x*-axis (horizontally to the right) and in a positive direction on the *y*-axis (vertically going down).



typeOf	**ModelShape**
name	**ProcessShape**
comment	
borderShape	
shapeIcon	
figure	ProcessFigure
childContainer	
portShape	

childContainer:

typeOf	**ChildContainer**
position	
size	

position:

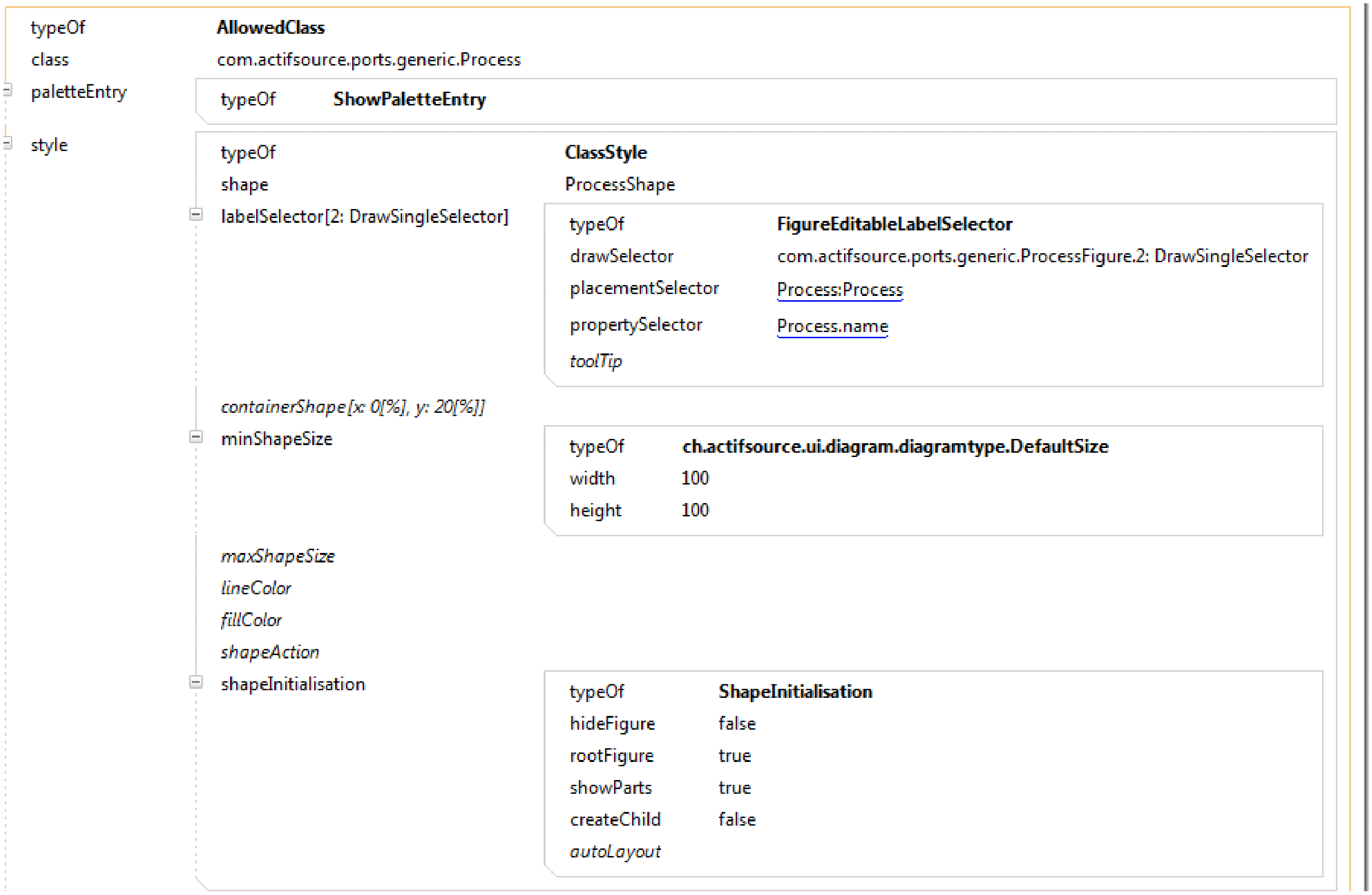
typeOf	**ch.actifsource.ui.diagram.figure.Point**
x	0
x unit	Percent
y	20
y unit	Percent

size:

typeOf	**ch.actifsource.ui.diagram.figure.Size**
width	100
width unit	Percent
height	100
height unit	Percent

We define a **childContainer** for the ProcessShape:

- Create a **ChildContainer**
- In the newly created **ChildContainer** define a **Point** as position with x = 0% and y = 20% (the container should not cover the uppermost rectangular section of the shape)
- Define a **Size** with width=100% and height=100%

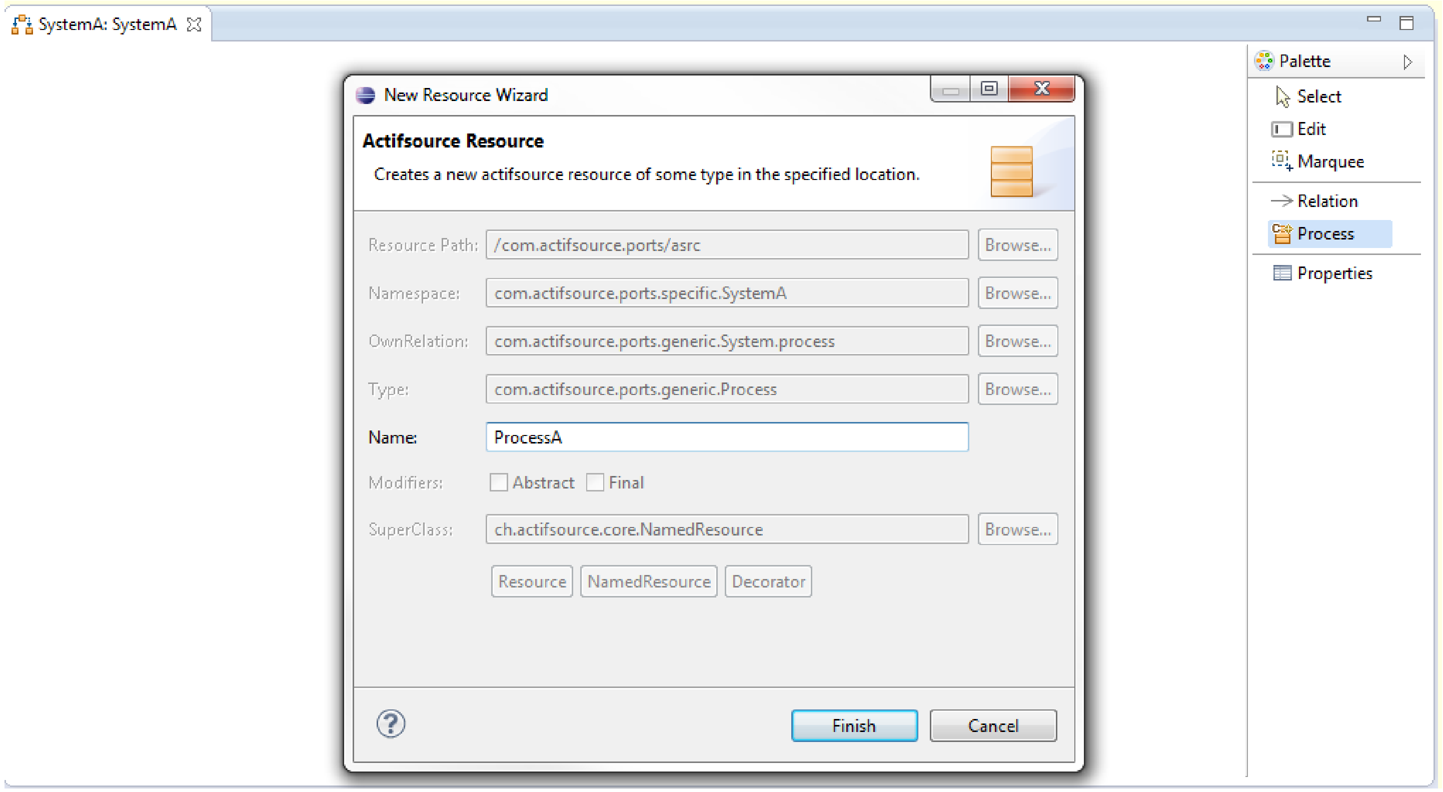


typeOf	**AllowedClass**		
class	com.actifsource.ports.generic.Process		
paletteEntry	typeOf	**ShowPaletteEntry**	
style	typeOf	**ClassStyle**	
	shape	ProcessShape	
	labelSelector[2: DrawSingleSelector]	typeOf	**FigureEditableLabelSelector**
		drawSelector	com.actifsource.ports.generic.ProcessFigure.2: DrawSingleSelector
		placementSelector	Process:Process
		propertySelector	Process.name
		toolTip	
	containerShape[x: 0[%], y: 20[%]]		
	minShapeSize	typeOf	**ch.actifsource.ui.diagram.diagramtype.DefaultSize**
		width	100
		height	100
	maxShapeSize		
	lineColor		
	fillColor		
	shapeAction		
	shapeInitialisation	typeOf	**ShapeInitialisation**
		hideFigure	false
		rootFigure	true
		showParts	true
		createChild	false
		autoLayout	

We define the label of a process shape to be the name of the corresponding Process, choose the initialization properties of a process shape and define a minimum size of process shapes:

- Create a **labelSelector** of type **FigureEditableLabelSelector** (this allows us to edit the name of a Process directly from the **Diagram Editor**)
- Create a **minShapeSize** with width = 100 and height = 100 (size in Pixels)
- Create a **ShapeInitialization** and change its **showParts** attribute to true (i.e., the shape will not hide its parts when newly created)

Define a new Diagram Type



Actifsource Resource

Creates a new actifsource resource of some type in the specified location.

Resource Path: /com.actifsource.ports/asrc

Namespace: com.actifsource.ports.specific.SystemA

OwnRelation: com.actifsource.ports.generic.System.process

Type: com.actifsource.ports.generic.Process

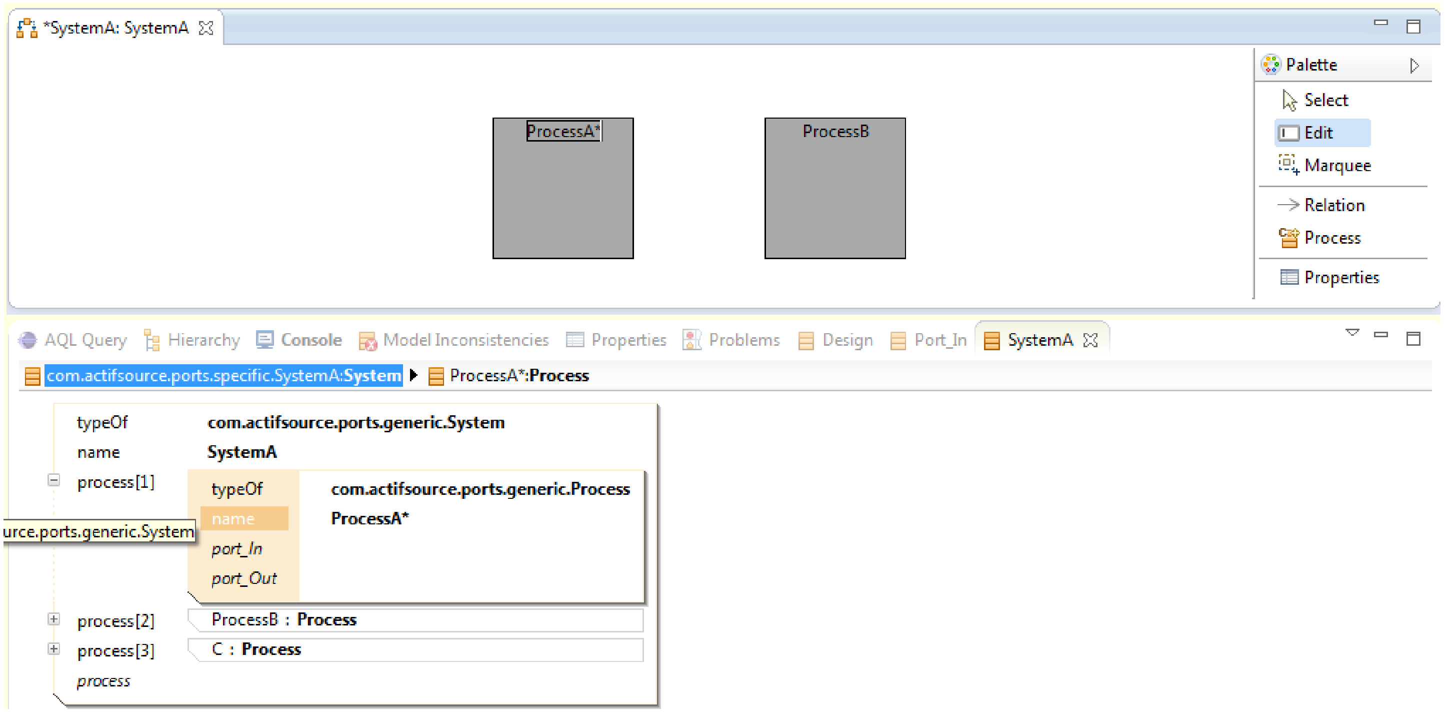
Name: ProcessA

Modifiers: [ ] Abstract [ ] Final

SuperClass: ch.actifsource.core.NamedResource

We create two Process instances, ProcessA and ProcessB, in the **Domain Diagram Editor**:

- Open the DomainDiagram SystemA in the **Diagram Editor**
- Select Process from the **Palette**
- Left-Click in the diagram to open the **New Resource Wizard** and choose `ProcessA` as the name of the new Process resource
- Repeat the previous step and choose `ProcessB` as the name of the second resource



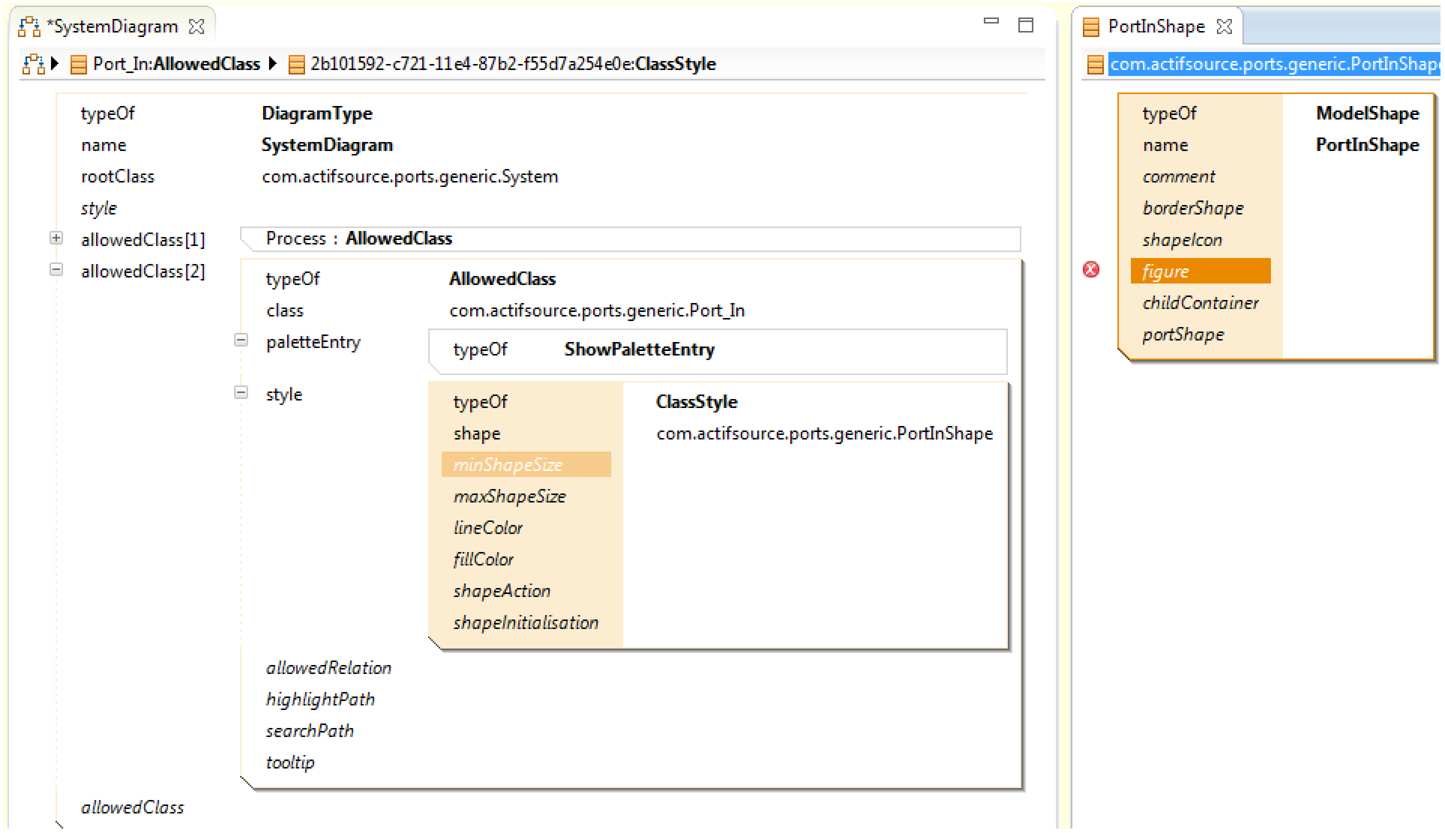
com.actifsource.ports.specific.SystemA:System ▸ ProcessA*:Process

typeOf com.actifsource.ports.generic.System
name SystemA
process[1]
 typeOf com.actifsource.ports.generic.Process
 name ProcessA*
 port_In
 port_Out
process[2] ProcessB : Process
process[3] C : Process
process

Check and inspect the newly created resources:

- Check that Ctrl+Click on the label of a Process opens the corresponding process in the **Resource Editor**
- Choose Edit from the Palette and edit the name of a process shape by left-clicking on its label. Note that the name in the resource editor is immediately updated when editing the label.

Define a new Diagram Type



*SystemDiagram
Port_In:AllowedClass ▸ 2b101592-c721-11e4-87b2-f55d7a254e0e:ClassStyle

typeOf	**DiagramType**
name	**SystemDiagram**
rootClass	com.actifsource.ports.generic.System
style	
allowedClass[1]	Process : **AllowedClass**
allowedClass[2]	

typeOf	**AllowedClass**
class	com.actifsource.ports.generic.Port_In
paletteEntry	typeOf **ShowPaletteEntry**
style	
allowedRelation	
highlightPath	
searchPath	
tooltip	

typeOf	**ClassStyle**
shape	com.actifsource.ports.generic.PortInShape
minShapeSize	
maxShapeSize	
lineColor	
fillColor	
shapeAction	
shapeInitialisation	

allowedClass

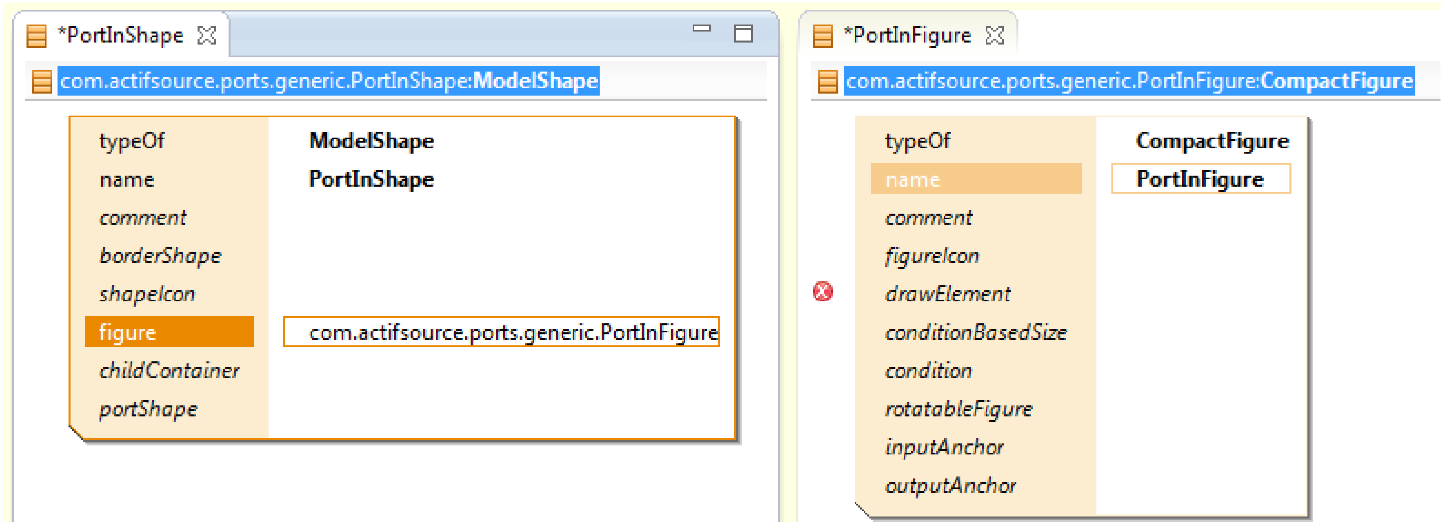
PortInShape
com.actifsource.ports.generic.PortInShap

typeOf	**ModelShape**
name	**PortInShape**
comment	
borderShape	
shapeIcon	
figure	
childContainer	
portShape	

Next, we define a shape and figure for Port_Ins:

- Add a new **AllowedClass** with class Port_In and define a **palleteEntry** of type **ShowPaletteEntry**
- Add a statement **style** and then create a new **shape** statement refering to a new **ModelShape**
- Give the name `PortInShape` to the new resource

Define a new Diagram Type


*PortInShape

com.actifsource.ports.generic.PortInShape:ModelShape

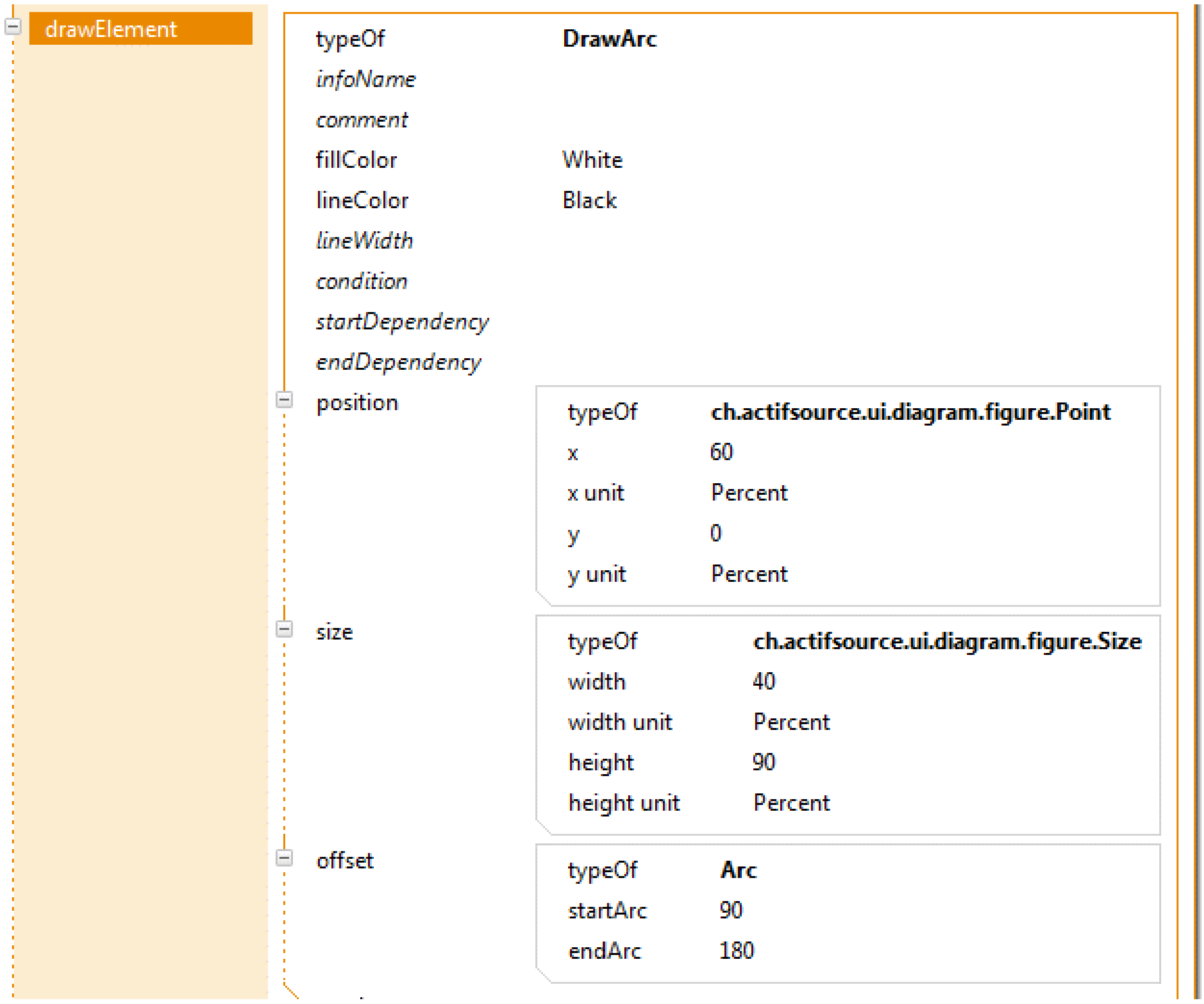
typeOf	ModelShape
name	PortInShape
comment	
borderShape	
shapeIcon	
figure	com.actifsource.ports.generic.PortInFigure
childContainer	
portShape	

*PortInFigure

com.actifsource.ports.generic.PortInFigure:CompactFigure

typeOf	CompactFigure
name	PortInFigure
comment	
figureIcon	
drawElement	
conditionBasedSize	
condition	
rotatableFigure	
inputAnchor	
outputAnchor	

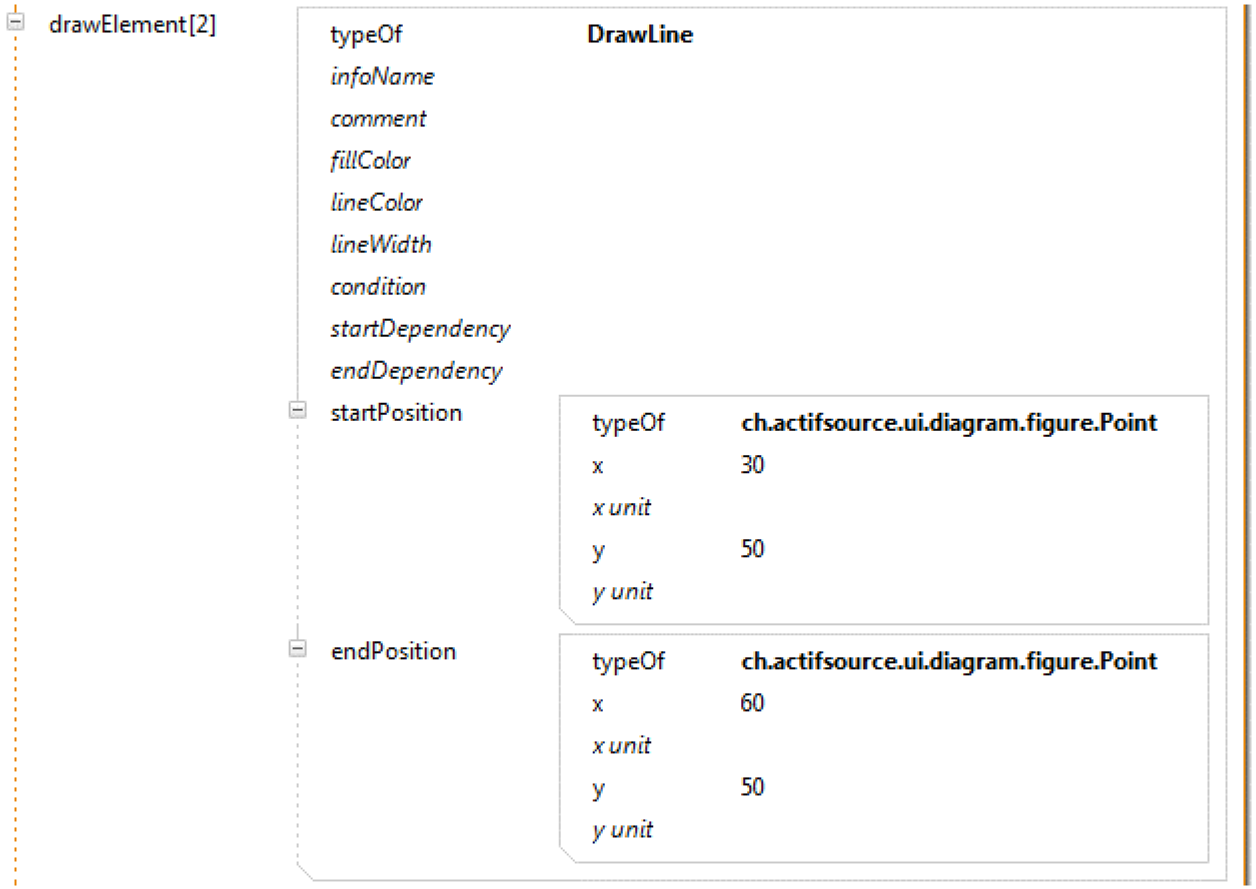
↳ Create a **figure** of type **CompactFigure** with the name `PortInFigure`


drawElement

typeOf	**DrawArc**
infoName	
comment	
fillColor	White
lineColor	Black
lineWidth	
condition	
startDependency	
endDependency	
position	typeOf **ch.actifsource.ui.diagram.figure.Point**; x 60; x unit Percent; y 0; y unit Percent
size	typeOf **ch.actifsource.ui.diagram.figure.Size**; width 40; width unit Percent; height 90; height unit Percent
offset	typeOf **Arc**; startArc 90; endArc 180

- Add a **drawElement** of type **DrawArc** to the PortInFigure
- Define the **fillColor** as **White** and the lineColor as **Black**
- Create a **position** with x=60% and y=0%
- Create a statement **size** refering to a resource of type **Size** with width=40% and height=90%
- Define an **offset** with startArc=90 and endArc=180 (degrees)
- ⓘ Note that 0° is positioned at the 3 o'clock position and positive values indicate a counter-clockwise rotation, negative values a clockwise rotation.

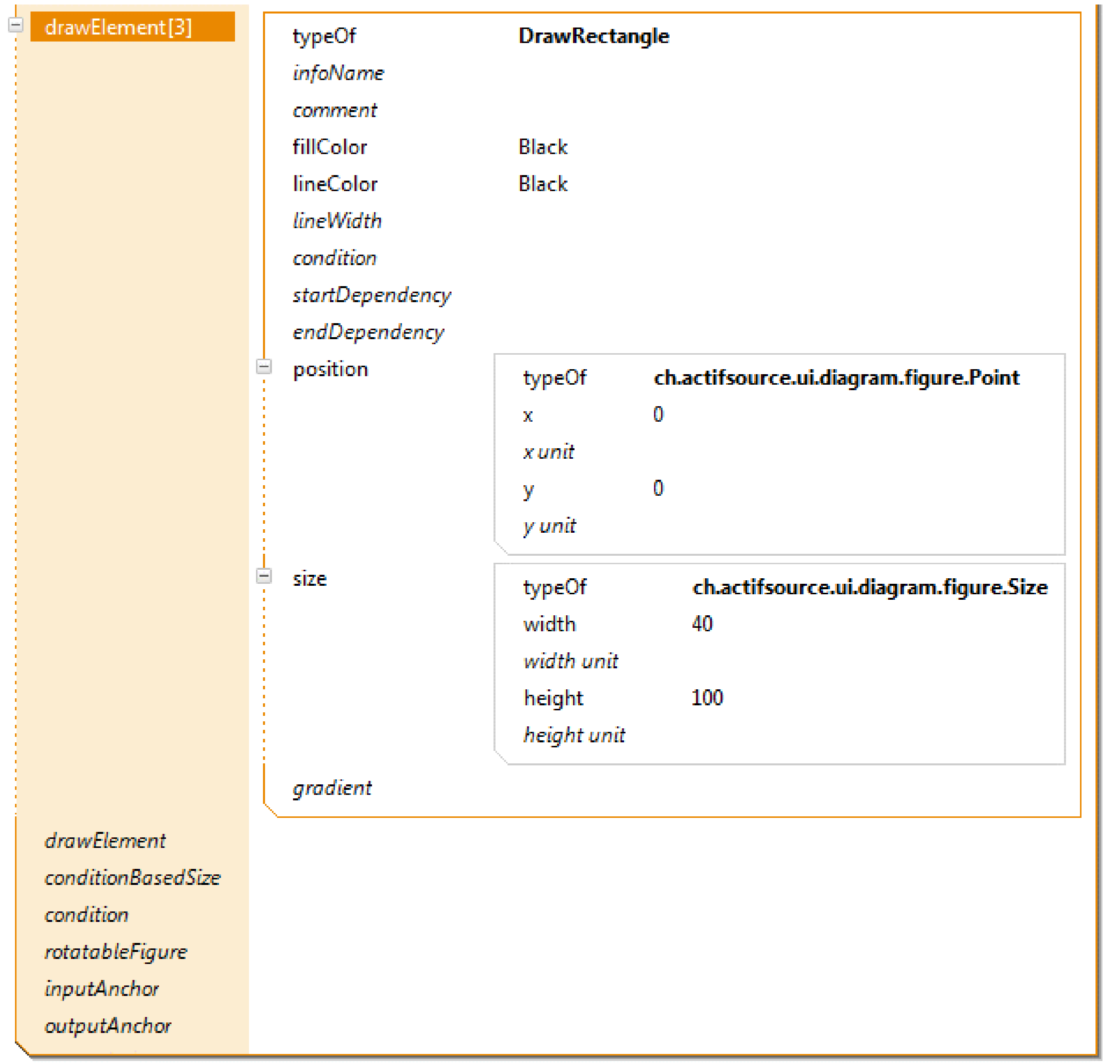
Define a new Diagram Type



drawElement[2]	typeOf	DrawLine	
	infoName		
	comment		
	fillColor		
	lineColor		
	lineWidth		
	condition		
	startDependency		
	endDependency		
	startPosition	typeOf	ch.actifsource.ui.diagram.figure.Point
		x	30
		x unit	
		y	50
		y unit	
	endPosition	typeOf	ch.actifsource.ui.diagram.figure.Point
		x	60
		x unit	
		y	50
		y unit	

- Add a second **drawElement** of type **DrawLine** to the PortInFigure
- Create a **startPosition** with x=30% and y=50%
- Create an **endPosition** with x=60% and y=50%

Define a new Diagram Type

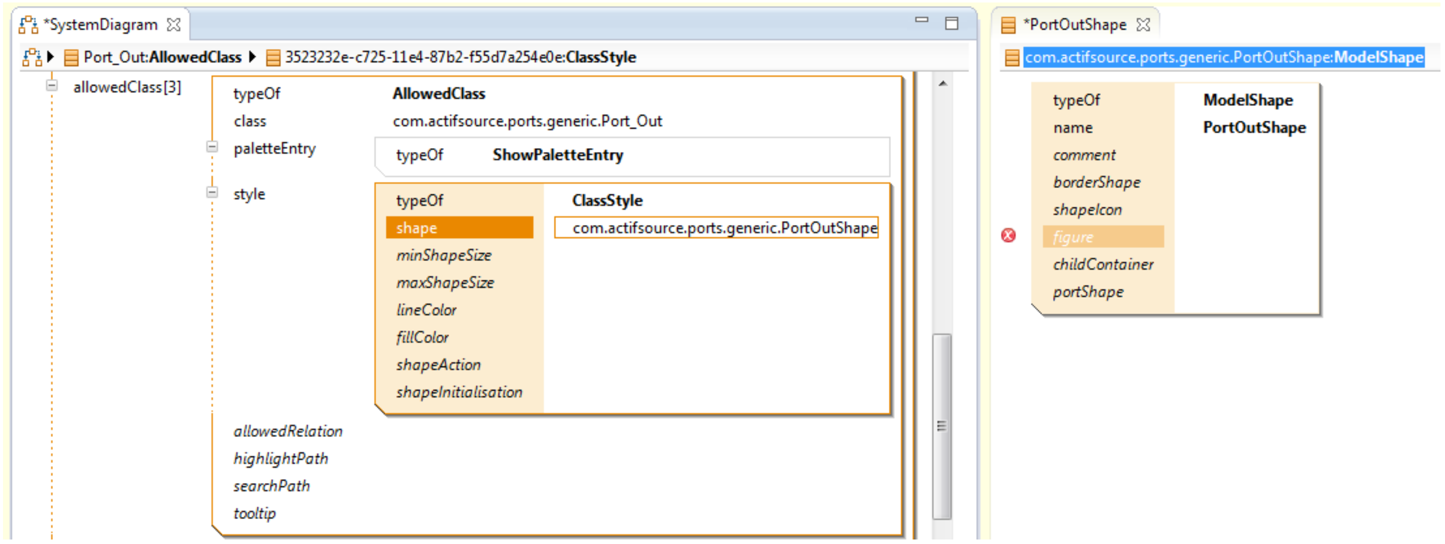

drawElement[3]

typeOf	**DrawRectangle**		
infoName			
comment			
fillColor	Black		
lineColor	Black		
lineWidth			
condition			
startDependency			
endDependency			
position		typeOf	**ch.actifsource.ui.diagram.figure.Point**
		x	0
		x unit	
		y	0
		y unit	
size		typeOf	**ch.actifsource.ui.diagram.figure.Size**
		width	40
		width unit	
		height	100
		height unit	
gradient			

drawElement
conditionBasedSize
condition
rotatableFigure
inputAnchor
outputAnchor

- Add a third **drawElement** of type **DrawRecangle** to the PortInFigure
- Create a statement **position** and set x=0% and y=0%
- Create a statement **size** and set width=40% and height=100%

Define a new Diagram Type

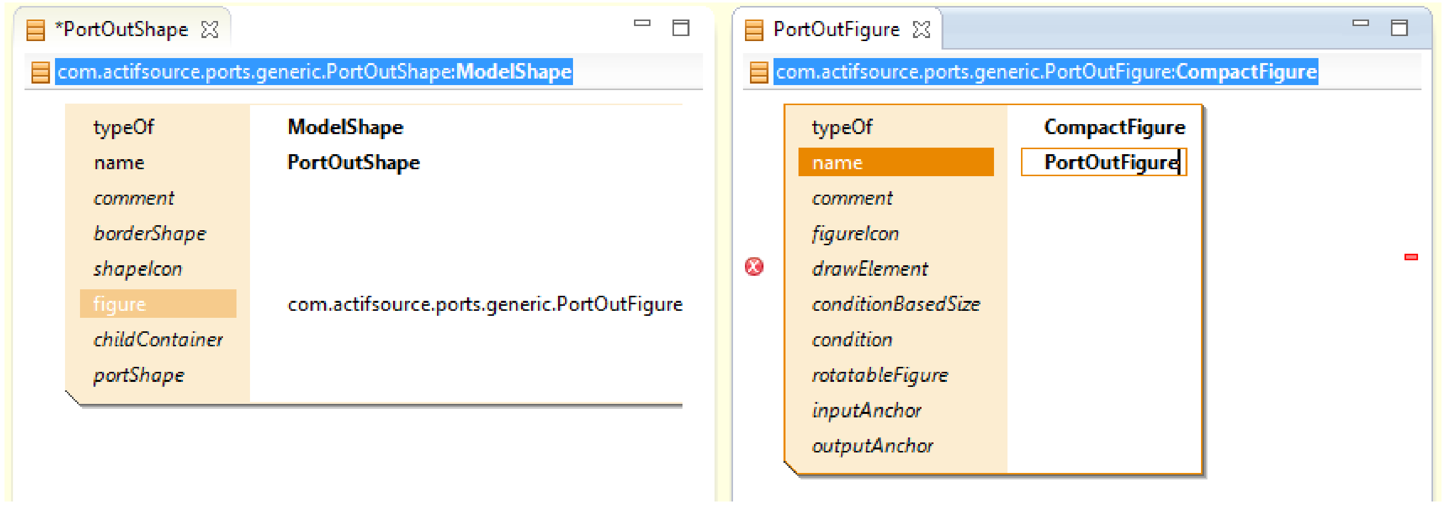

*SystemDiagram
Port_Out:AllowedClass
3523232e-c725-11e4-87b2-f55d7a254e0e:ClassStyle
allowedClass[3]
*PortOutShape
com.actifsource.ports.generic.PortOutShape:ModelShape

typeOf	AllowedClass
class	com.actifsource.ports.generic.Port_Out
paletteEntry	typeOf ShowPaletteEntry
style	typeOf ClassStyle; shape com.actifsource.ports.generic.PortOutShape; minShapeSize; maxShapeSize; lineColor; fillColor; shapeAction; shapeInitialisation
allowedRelation	
highlightPath	
searchPath	
tooltip	

typeOf	ModelShape
name	PortOutShape
comment	
borderShape	
shapeIcon	
figure	
childContainer	
portShape	

- Open SystemDiagram in the **Resource Editor**
- Create a new **AllowedClass** that refers to the class Port_Out
- Create a new **ClassStyle** that refers to a new **ModelShape**. Give the name `PortOutShape` to the newly created **ModelShape**.

Define a new Diagram Type


*PortOutShape
com.actifsource.ports.generic.PortOutShape:**ModelShape**

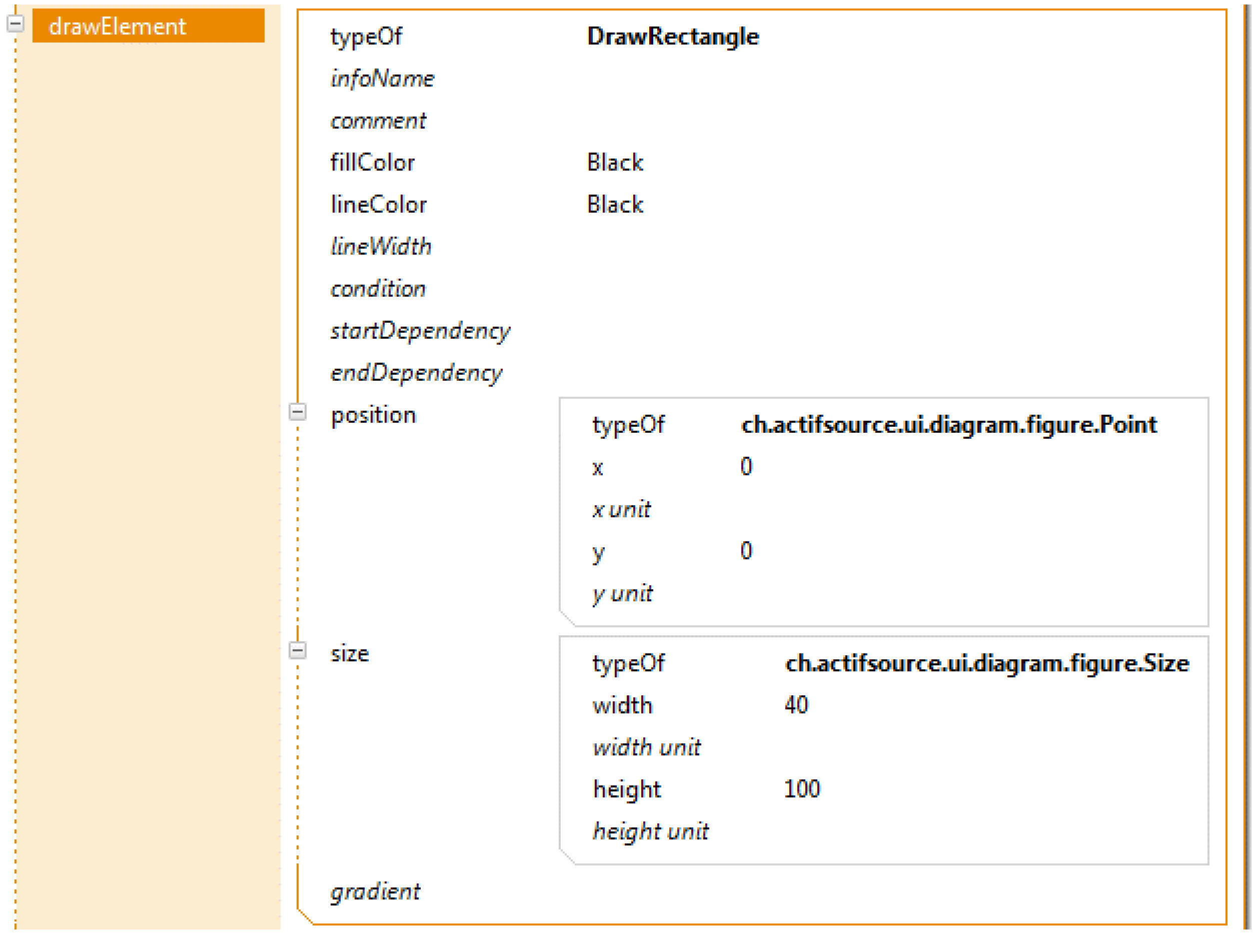
typeOf	**ModelShape**
name	**PortOutShape**
comment	
borderShape	
shapeIcon	
figure	com.actifsource.ports.generic.PortOutFigure
childContainer	
portShape	

PortOutFigure
com.actifsource.ports.generic.PortOutFigure:**CompactFigure**

typeOf	**CompactFigure**
name	**PortOutFigure**
comment	
figureIcon	
drawElement	
conditionBasedSize	
condition	
rotatableFigure	
inputAnchor	
outputAnchor	

- Add a **<u>CompactFigure</u>** with the name `PortOutFigure` to <u>PortOutShape</u> as shown above

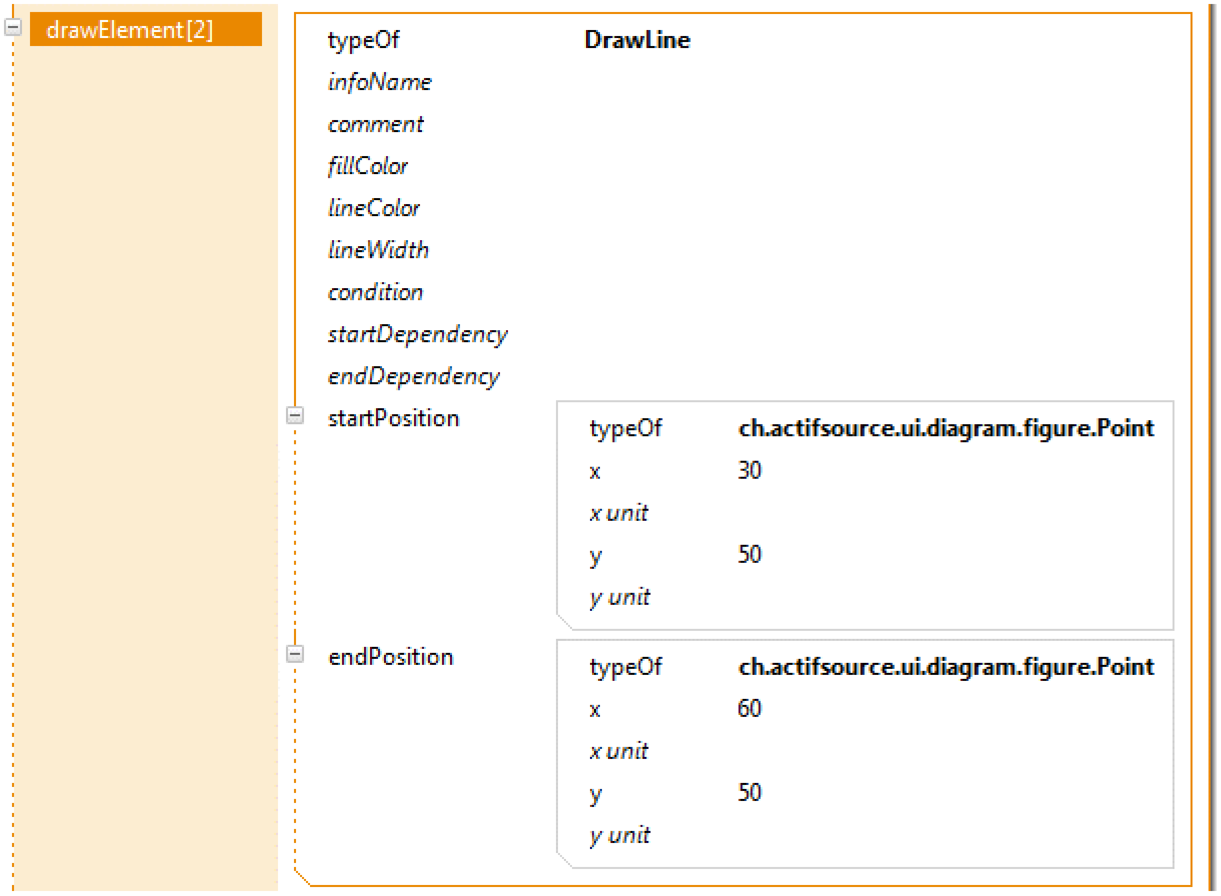
Define a new Diagram Type


drawElement

typeOf	**DrawRectangle**
infoName	
comment	
fillColor	Black
lineColor	Black
lineWidth	
condition	
startDependency	
endDependency	
position	typeOf **ch.actifsource.ui.diagram.figure.Point** x 0 *x unit* y 0 *y unit*
size	typeOf **ch.actifsource.ui.diagram.figure.Size** width 40 *width unit* height 100 *height unit*
gradient	

- Add a **drawElement** of type DrawRecangle to the PortOutFigure
- Create a **statement** position and set x=0% and y=0%
- Create a statement **size** and set width=40% and height=100%

Define a new Diagram Type


drawElement[2]

typeOf	DrawLine
infoName	
comment	
fillColor	
lineColor	
lineWidth	
condition	
startDependency	
endDependency	
startPosition	
endPosition	

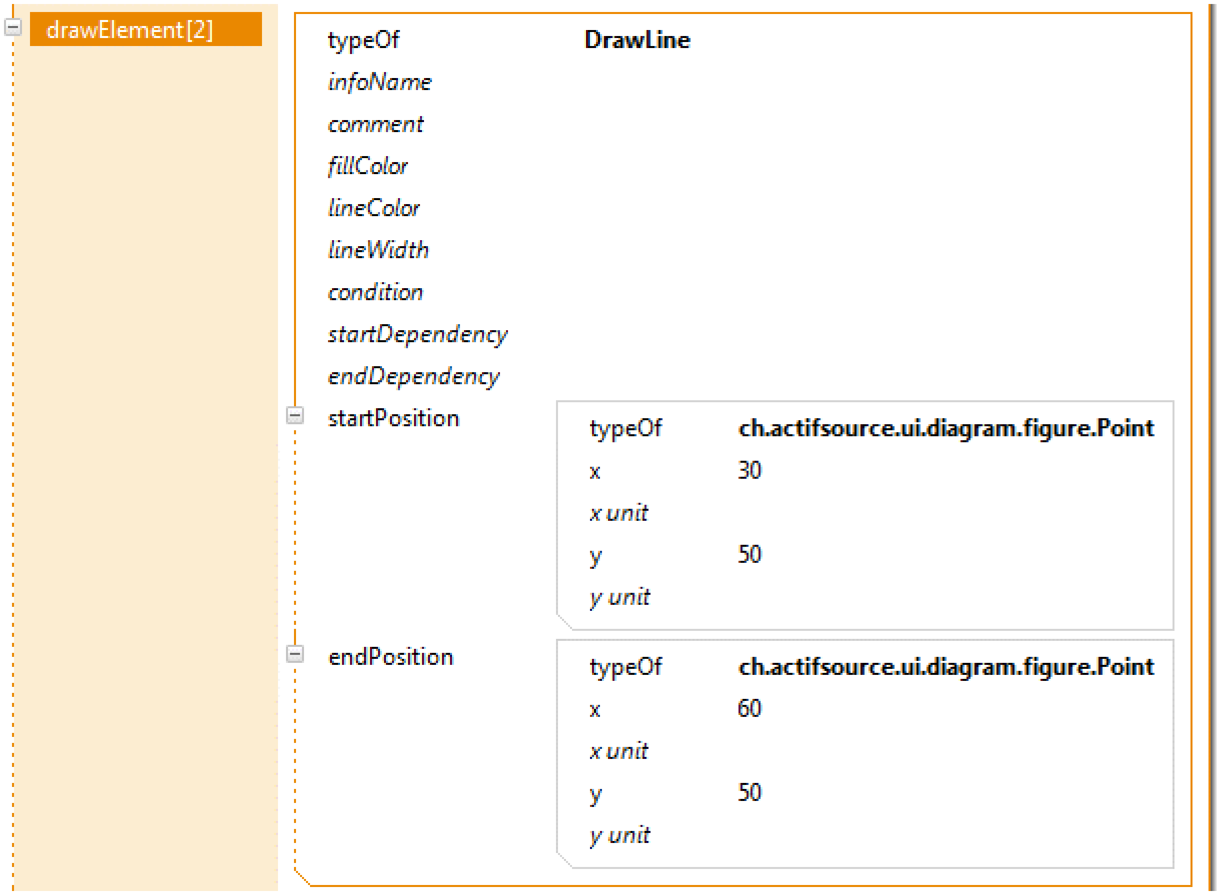
startPosition:

typeOf	ch.actifsource.ui.diagram.figure.Point
x	30
x unit	
y	50
y unit	

endPosition:

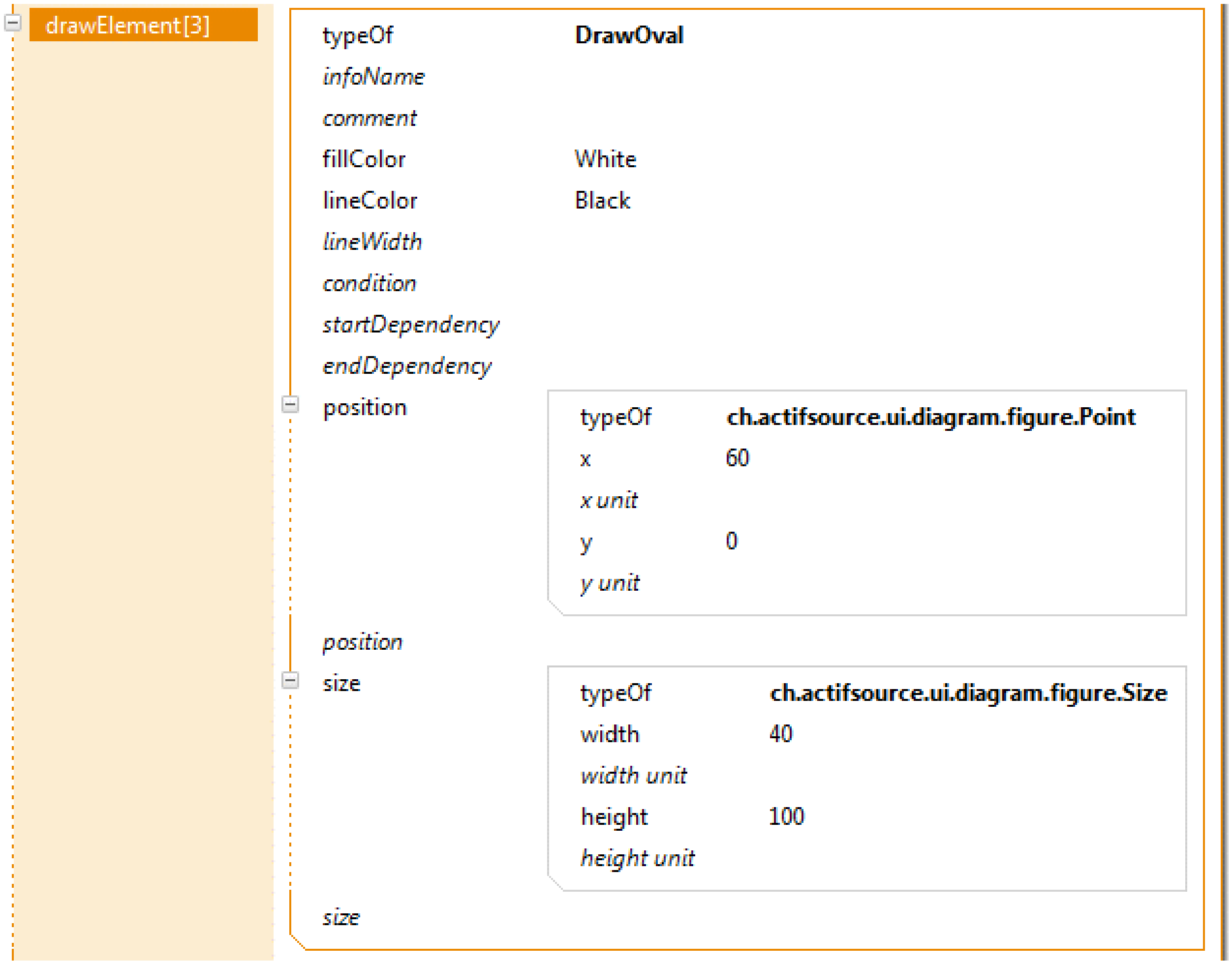
typeOf	ch.actifsource.ui.diagram.figure.Point
x	60
x unit	
y	50
y unit	

- Add a second **drawElement** of type **DrawLine** to the PortInFigure
- Create a **startPosition** with x=30% and y=50%
- Create an **endPosition** with x=60% and y=50%



typeOf	DrawLine
infoName	
comment	
fillColor	
lineColor	
lineWidth	
condition	
startDependency	
endDependency	
startPosition	typeOf: ch.actifsource.ui.diagram.figure.Point; x: 30; *x unit*; y: 50; *y unit*
endPosition	typeOf: ch.actifsource.ui.diagram.figure.Point; x: 60; *x unit*; y: 50; *y unit*

- Add a second **drawElement** of type **DrawLine** to the PortInFigure
- Create a **startPosition** with x=30% and y=50%
- Create an **endPosition** with x=60% and y=50%


drawElement[3]

typeOf	DrawOval
infoName	
comment	
fillColor	White
lineColor	Black
lineWidth	
condition	
startDependency	
endDependency	
position	
position	
size	
size	

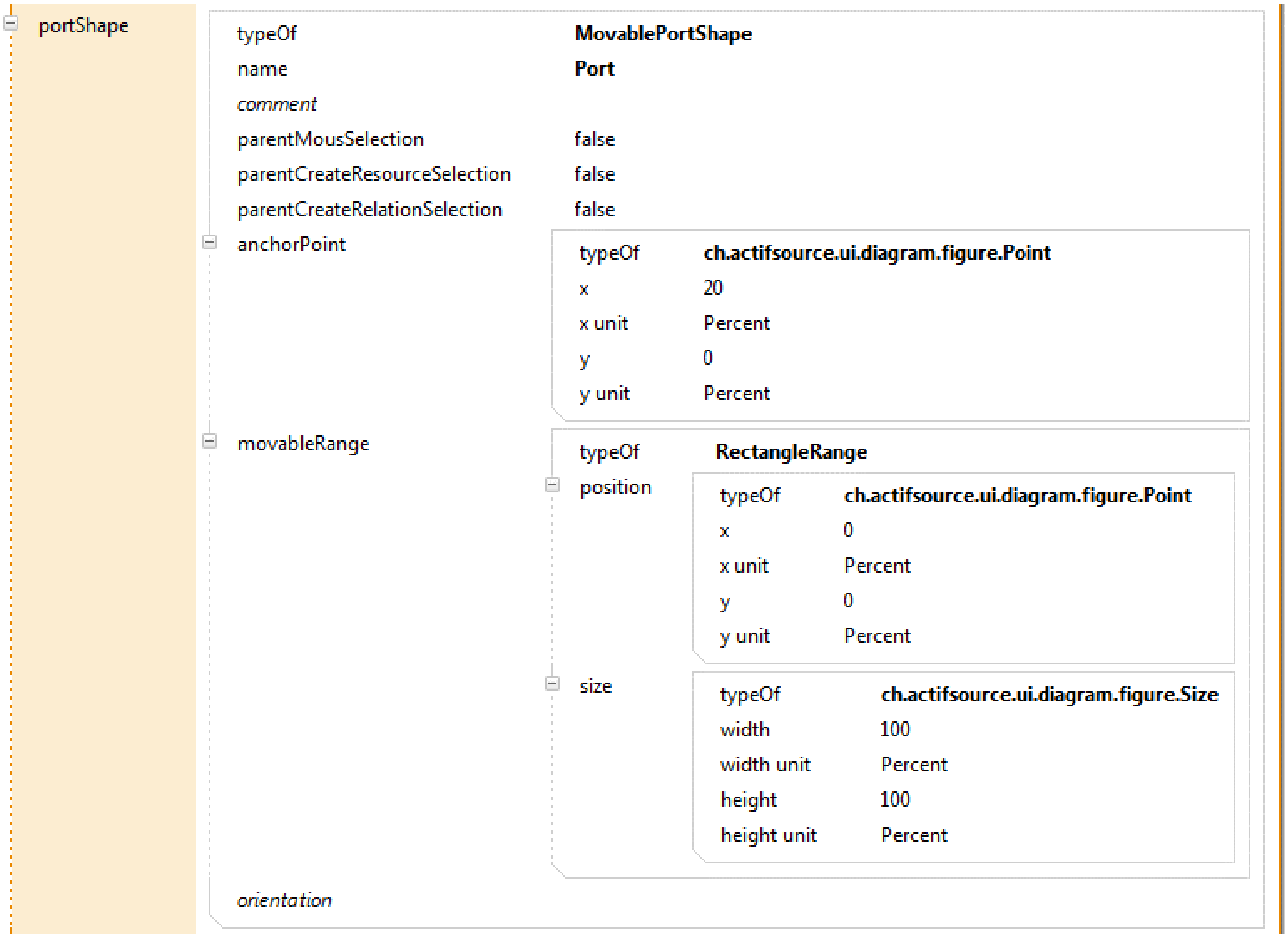
position:

typeOf	ch.actifsource.ui.diagram.figure.Point
x	60
x unit	
y	0
y unit	

size:

typeOf	ch.actifsource.ui.diagram.figure.Size
width	40
width unit	
height	100
height unit	

- Add a third **drawElement** of type **DrawOval** to the PortOutFigure
- Define the **fillColor** as **White** and the lineColor as **Black**
- Define the **position** with x=60% and y=0%
- Define the **size** with width=40% and height=100%

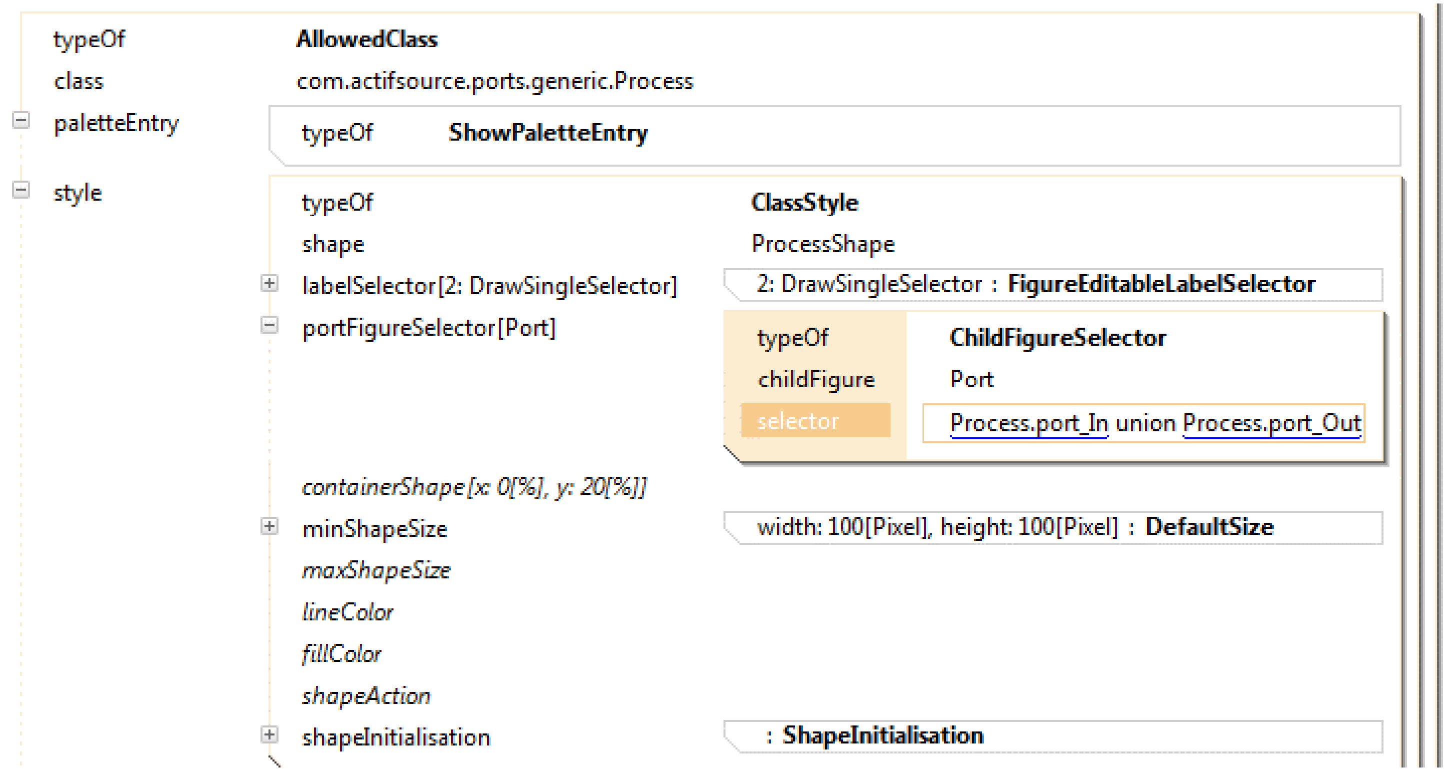


portShape

typeOf	**MovablePortShape**
name	**Port**
comment	
parentMousSelection	false
parentCreateResourceSelection	false
parentCreateRelationSelection	false
anchorPoint	typeOf **ch.actifsource.ui.diagram.figure.Point**; x 20; x unit Percent; y 0; y unit Percent
movableRange	typeOf **RectangleRange**; position: typeOf **ch.actifsource.ui.diagram.figure.Point**, x 0, x unit Percent, y 0, y unit Percent; size: typeOf **ch.actifsource.ui.diagram.figure.Size**, width 100, width unit Percent, height 100, height unit Percent
orientation	

We define the position of ports relative to their process shapes and their behavior:

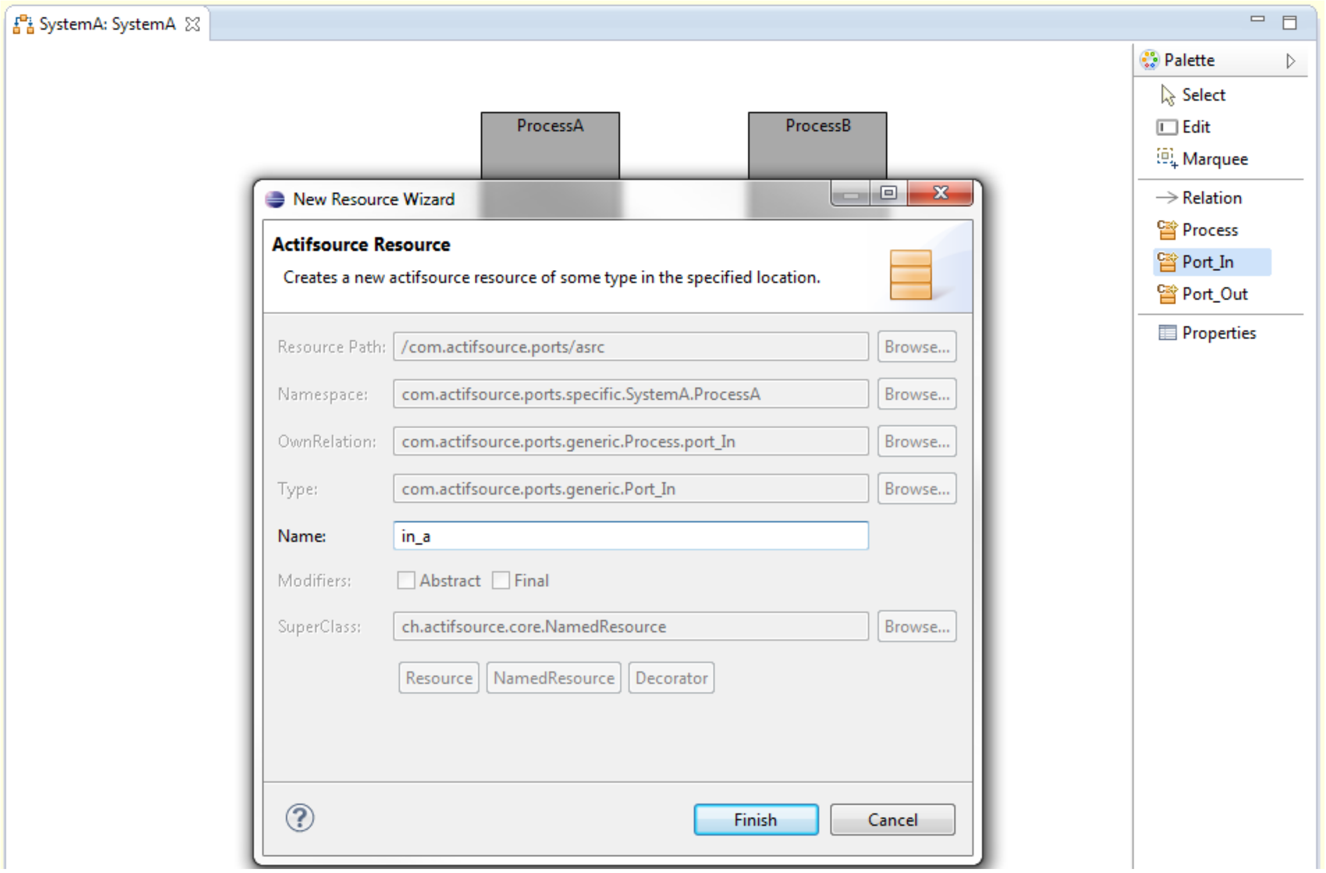
- Create a **MovablePortShape** in the ProcessShape
- Define an **anchorPoint** with x=20% and y=0%
- Define a **movableRange** with a **position** with x=0% and y=0% and a **size** with width=100% and height=100% (meaning that a port can be moved to any point on the boundary of the rectangular process shape)
- We do not define an **orientation** (the shape of a port is thus rotated when moved along the boundary of its parent shape)



typeOf	**AllowedClass**
class	com.actifsource.ports.generic.Process
paletteEntry	typeOf **ShowPaletteEntry**
style	typeOf **ClassStyle**
	shape ProcessShape
	labelSelector[2: DrawSingleSelector] 2: DrawSingleSelector : **FigureEditableLabelSelector**
	portFigureSelector[Port] typeOf **ChildFigureSelector**; childFigure Port; selector Process.port_In union Process.port_Out
	containerShape[x: 0[%], y: 20[%]]
	minShapeSize width: 100[Pixel], height: 100[Pixel] : **DefaultSize**
	maxShapeSize
	lineColor
	fillColor
	shapeAction
	shapeInitialisation : **ShapeInitialisation**

We define a selector for all the allowed Port figures:

- Open the SystemDiagram in the **Resource Editor** and create a **ChildFigureSelector** as **Decorator** for portFigureSelector[Port]
- Define the selector Process.port_In union Process.port_Out which selects all ports of type Port_In and Port_Out that belong to a Process



Actifsource Resource

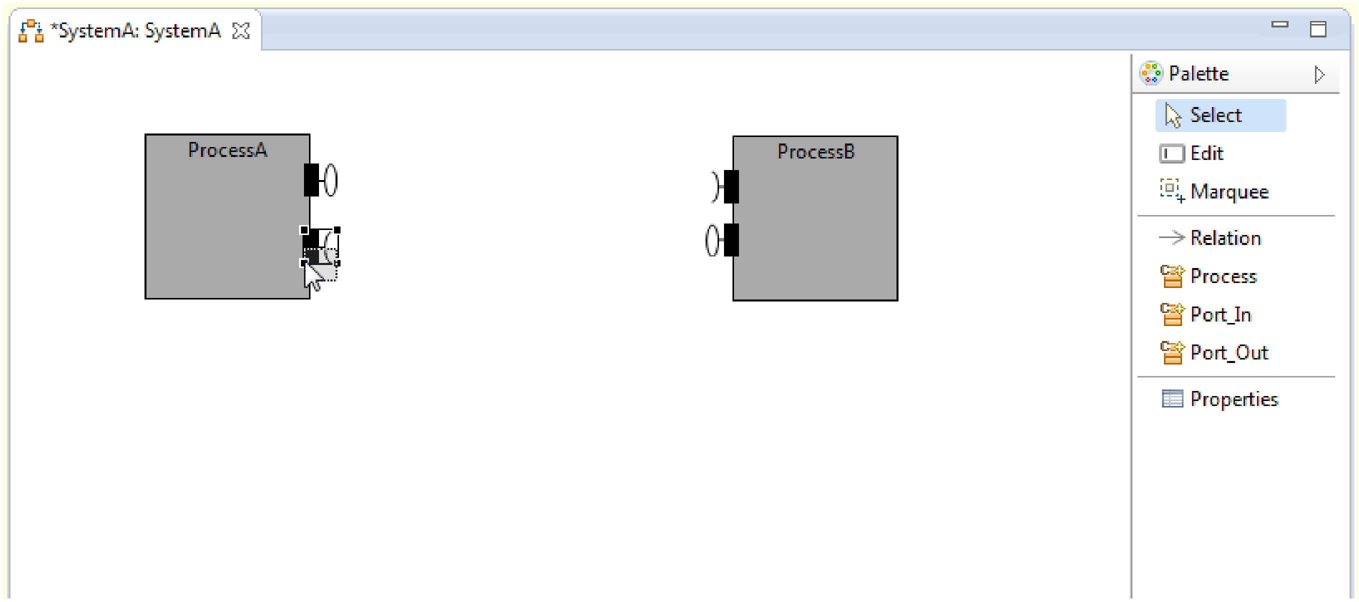
Creates a new actifsource resource of some type in the specified location.

Resource Path:	/com.actifsource.ports/asrc
Namespace:	com.actifsource.ports.specific.SystemA.ProcessA
OwnRelation:	com.actifsource.ports.generic.Process.port_In
Type:	com.actifsource.ports.generic.Port_In
Name:	in_a
Modifiers:	Abstract, Final
SuperClass:	ch.actifsource.core.NamedResource

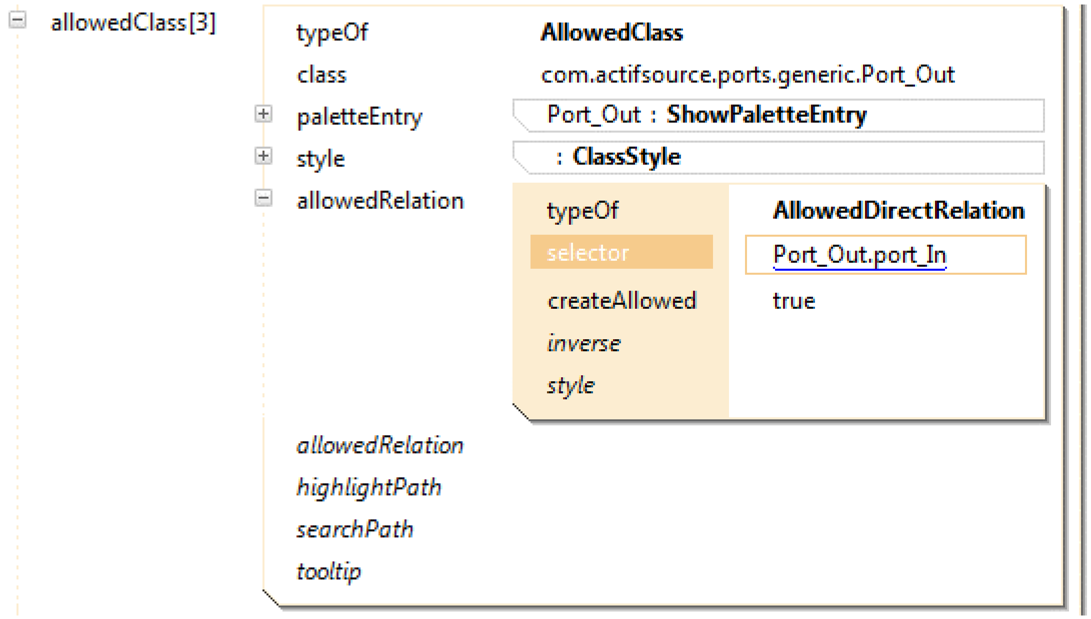
Resource NamedResource Decorator

Next, we a PortIn and PortOut to each of our two Processes:

- Open the Domain Diagram SystemA in the **Diagram Editor** (**Note:** ***If the diagram is still open, close it first and then re-open it in order to update the behavior of the diagram***)
- Choose Port_In from the Palette and left-click on ProcessA
- Choose the name *in_a* in the opened **New Resource Wizard**
- In the same way create a Port_In with name *in_b* for ProcessB
- Choose Port_Out from the Palette and left-click on ProcessA
- Choose the name *out_a* in the opened **New Resource Wizard**
- In the same way create a Port_In with name *out_b* for ProcessB



↳ Open the Domain Diagram SystemA in the **Diagram Editor** and position the newly created ports by selecting them and moving them on the boundary of their process shapes.



allowedClass[3]

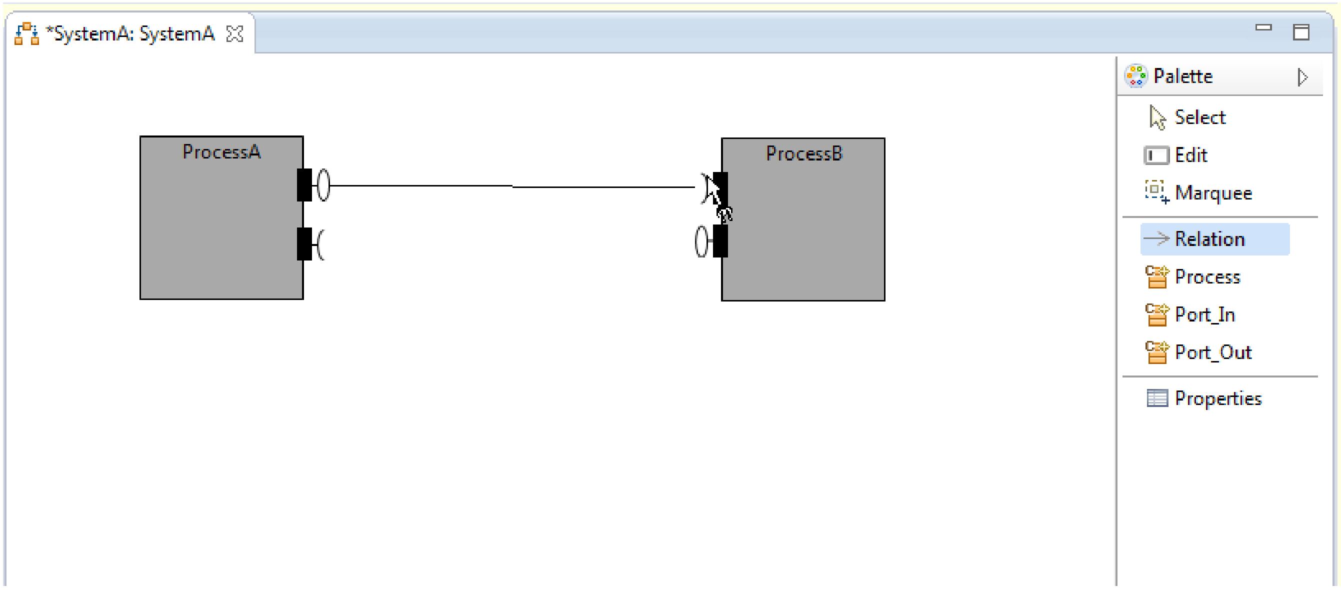
typeOf	**AllowedClass**
class	com.actifsource.ports.generic.Port_Out
paletteEntry	Port_Out : **ShowPaletteEntry**
style	: **ClassStyle**
allowedRelation	
allowedRelation	
highlightPath	
searchPath	
tooltip	

allowedRelation:

typeOf	**AllowedDirectRelation**
selector	Port_Out.port_In
createAllowed	true
inverse	
style	

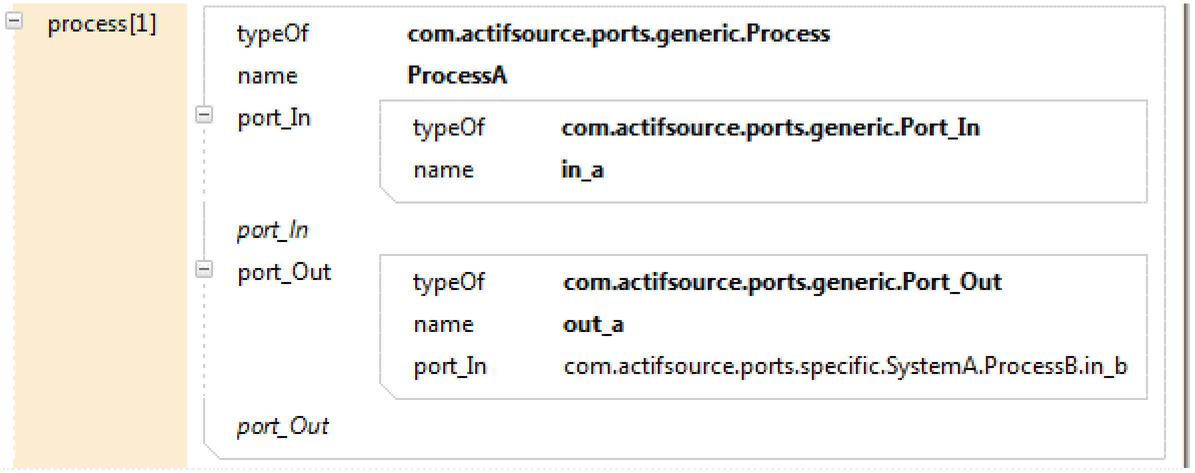
Finally, we want to be able to connect outgoing ports (Port_Out) to ingoing ports (Port_In) by the relation Port_Out.port_In:

- Open the SystemDiagram in the **Resource Editor**
- In the **allowedClass** for Port_Out, create an **allowedRelation** of type **AllowedDirectRelation** and define the selector Port_Out.port_In



We connect out_a with in_b as follows:

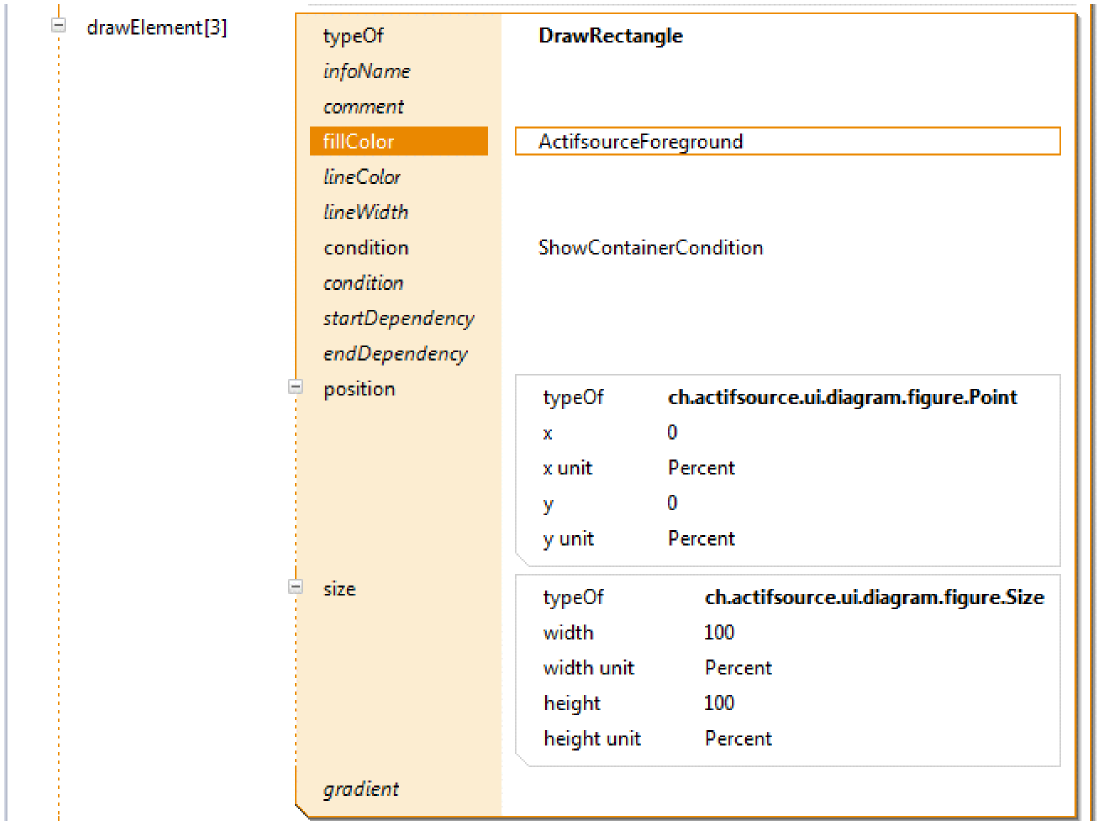
- Choose Relation from the Palette
- Click on the port out_a. Then drag the mouse to the port in_b and click on port in_b
- In the same way connect out_b to in_a
- Open and inspect the newly created statements by opening SystemA in the **Resource Editor** as well:



process[1]

typeOf	com.actifsource.ports.generic.Process		
name	ProcessA		
port_In	typeOf	com.actifsource.ports.generic.Port_In	
	name	in_a	
port_In			
port_Out	typeOf	com.actifsource.ports.generic.Port_Out	
	name	out_a	
	port_In	com.actifsource.ports.specific.SystemA.ProcessB.in_b	
port_Out			

Part III: Add conditions to figures



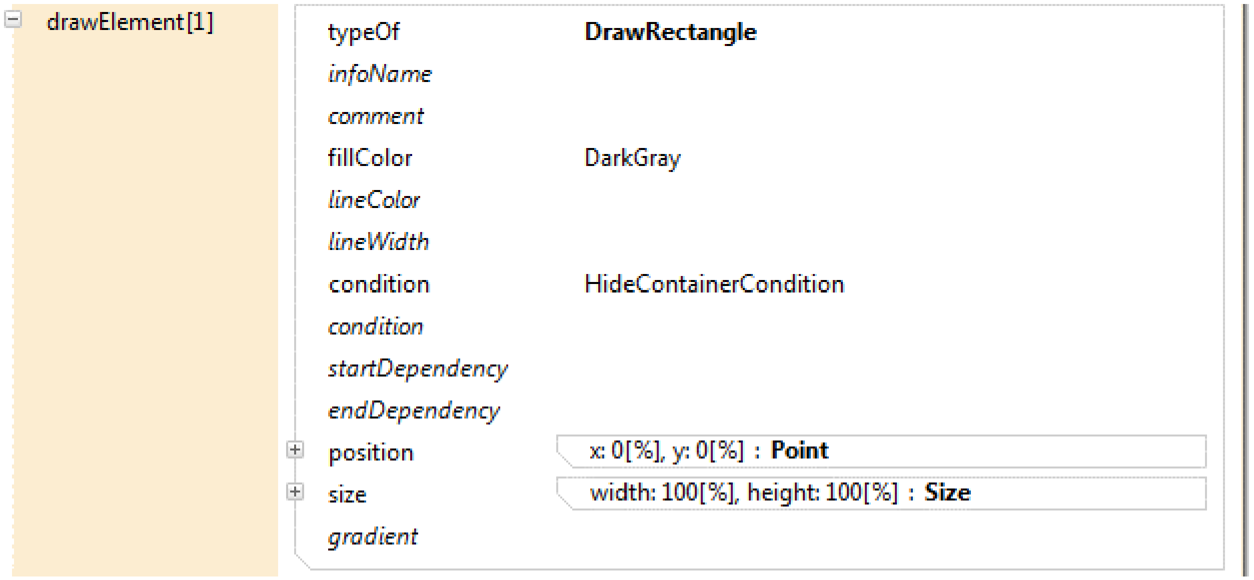
drawElement[3]

typeOf	**DrawRectangle**
infoName	
comment	
fillColor	ActifsourceForeground
lineColor	
lineWidth	
condition	ShowContainerCondition
condition	
startDependency	
endDependency	
position	typeOf **ch.actifsource.ui.diagram.figure.Point**; x 0; x unit Percent; y 0; y unit Percent
size	typeOf **ch.actifsource.ui.diagram.figure.Size**; width 100; width unit Percent; height 100; height unit Percent
gradient	

Next, we want the Process shape to **change** its color when its parts are hidden:

- Open ProcessFigure in the **Resource Editor** and add a third **drawElement** of type **DrawRectangle** and define the properties **size** and **position** as above (Note that you can also simply copy and paste the existing **DrawRectangle**).
- Define the **fillColor** as ActifsourceForeground
- Create a condition by using the Content Assist and choosing **ShowContainerCondition** (i.e., the shape should have the color **ActifsourceForeground** when its container is not hidden)

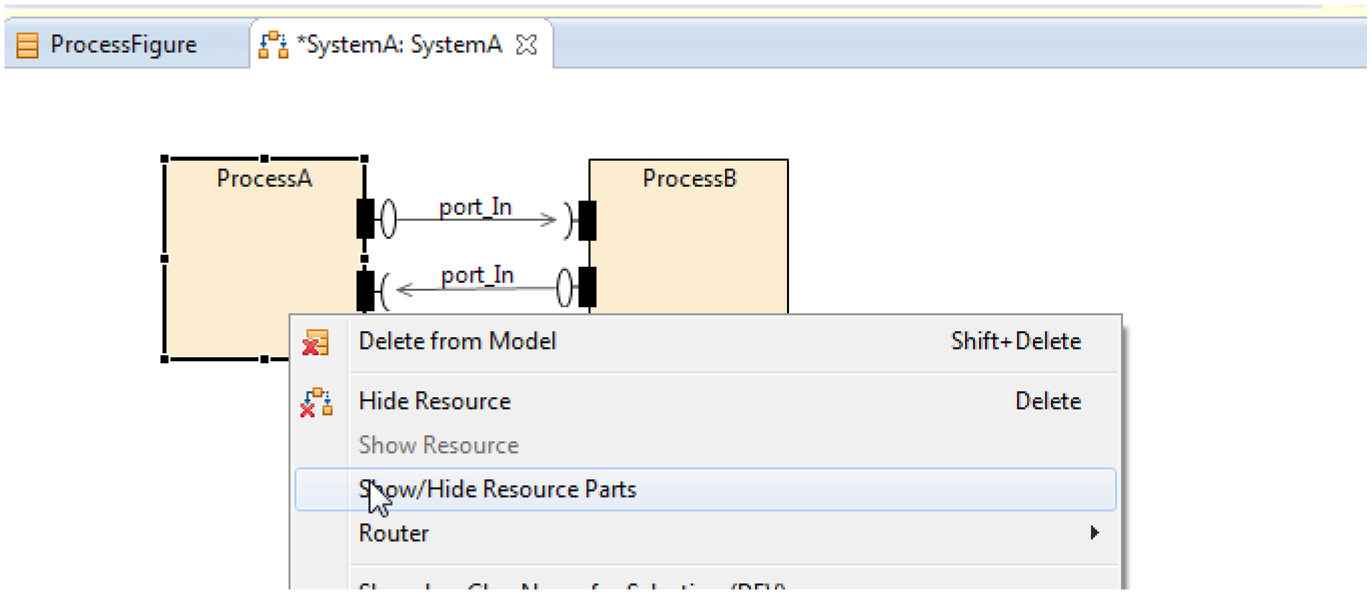
Add conditions to figures



drawElement[1]	typeOf	**DrawRectangle**
	infoName	
	comment	
	fillColor	DarkGray
	lineColor	
	lineWidth	
	condition	HideContainerCondition
	condition	
	startDependency	
	endDependency	
	position	x: 0[%], y: 0[%] : **Point**
	size	width: 100[%], height: 100[%] : **Size**
	gradient	

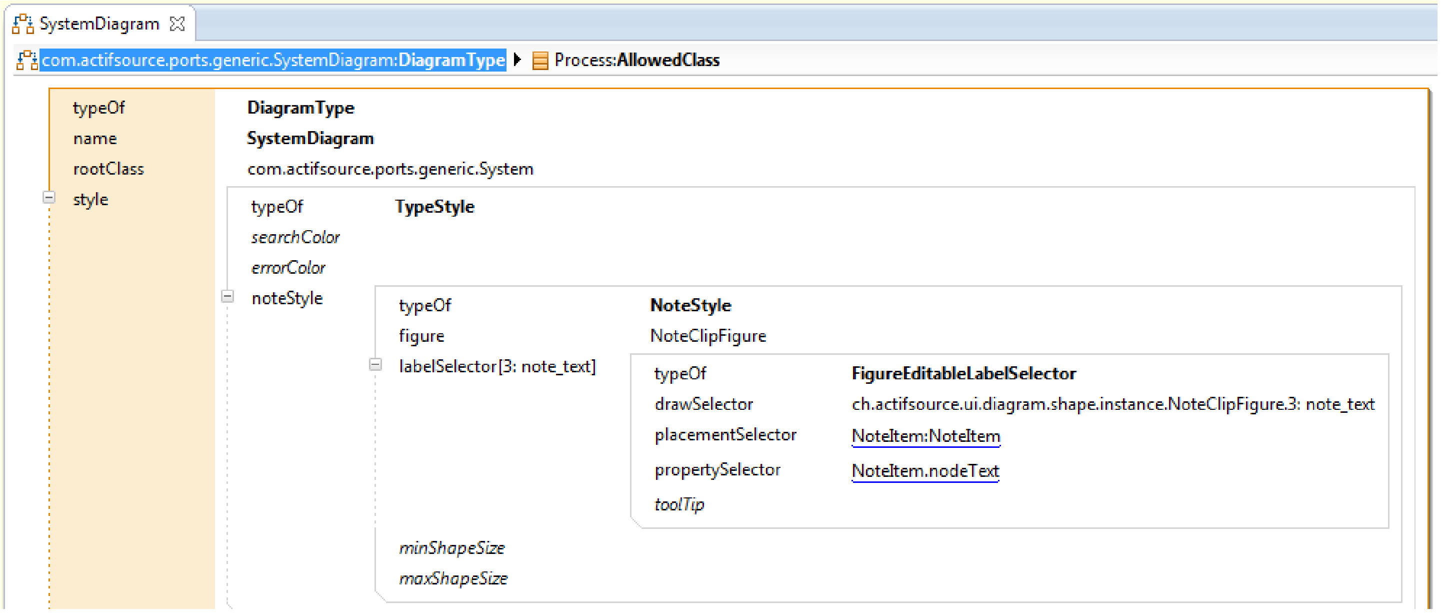
↳ Add a condition **HideContainerCondition** to the first drawElement of type **DrawRectangle** of the ProcessFigure (thus, the shape will be **DarkGray** when its container is hidden).

Add conditions to figures



- Open the Domain Diagram SystemA in the **Diagram Editor** and righ-click on one of the Process shapes
- Select **Show/Hide Resource Parts** from the menu and check that the color of the shape changes accordingly

Part IV: Add notes to Domain Diagrams



typeOf	**DiagramType**
name	**SystemDiagram**
rootClass	com.actifsource.ports.generic.System
style	

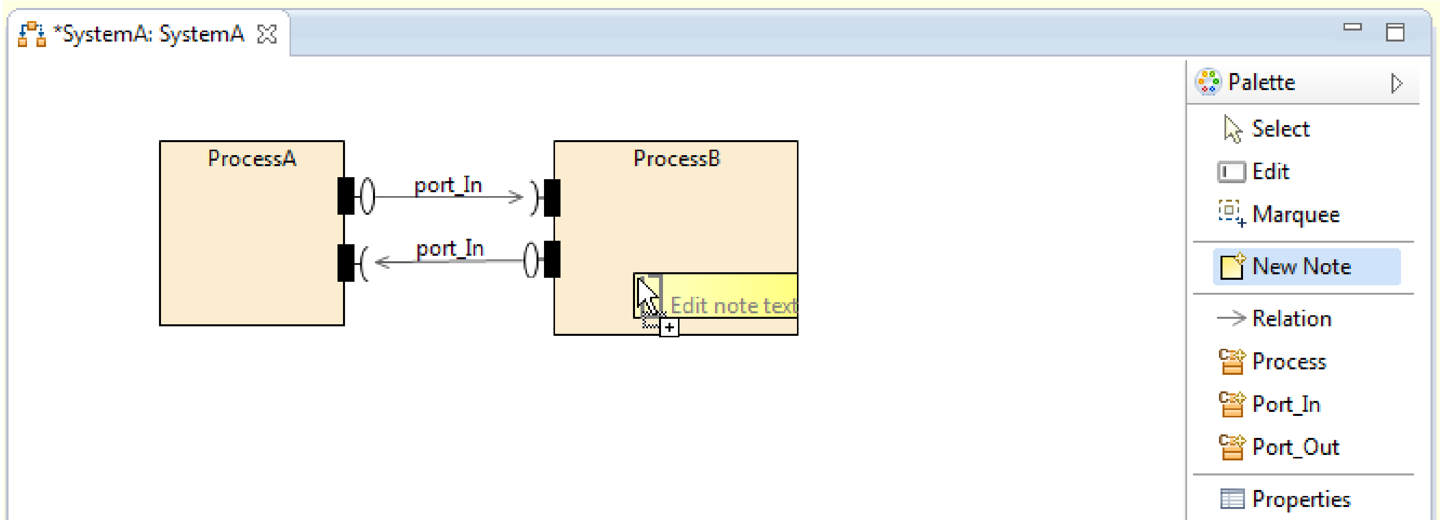
typeOf	**TypeStyle**
searchColor	
errorColor	
noteStyle	

typeOf	**NoteStyle**
figure	NoteClipFigure
labelSelector[3: note_text]	
minShapeSize	
maxShapeSize	

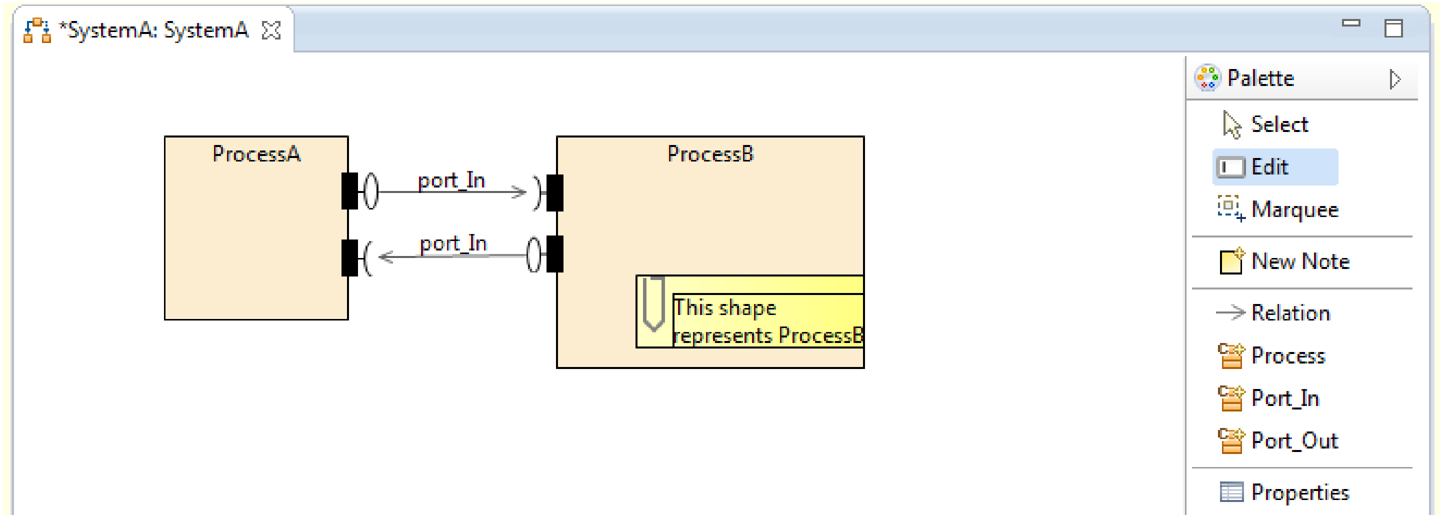
typeOf	**FigureEditableLabelSelector**
drawSelector	ch.actifsource.ui.diagram.shape.instance.NoteClipFigure.3: note_text
placementSelector	NoteItem:NoteItem
propertySelector	NoteItem.nodeText
toolTip	

- Open the SystemDiagram in the **Resource Editor**
- Create a **TypeStyle**
- Create a **NoteStyle** for the new **TypeStyle**
- Choose **NoteClipFigure** as figure
- Define a labelSelector of type **FigureEditableLabelSelector** and add the **placementSelector** NoteItem:NoteItem and the **propertySelector** NoteItem.nodeText

Add notes to Domain Diagrams

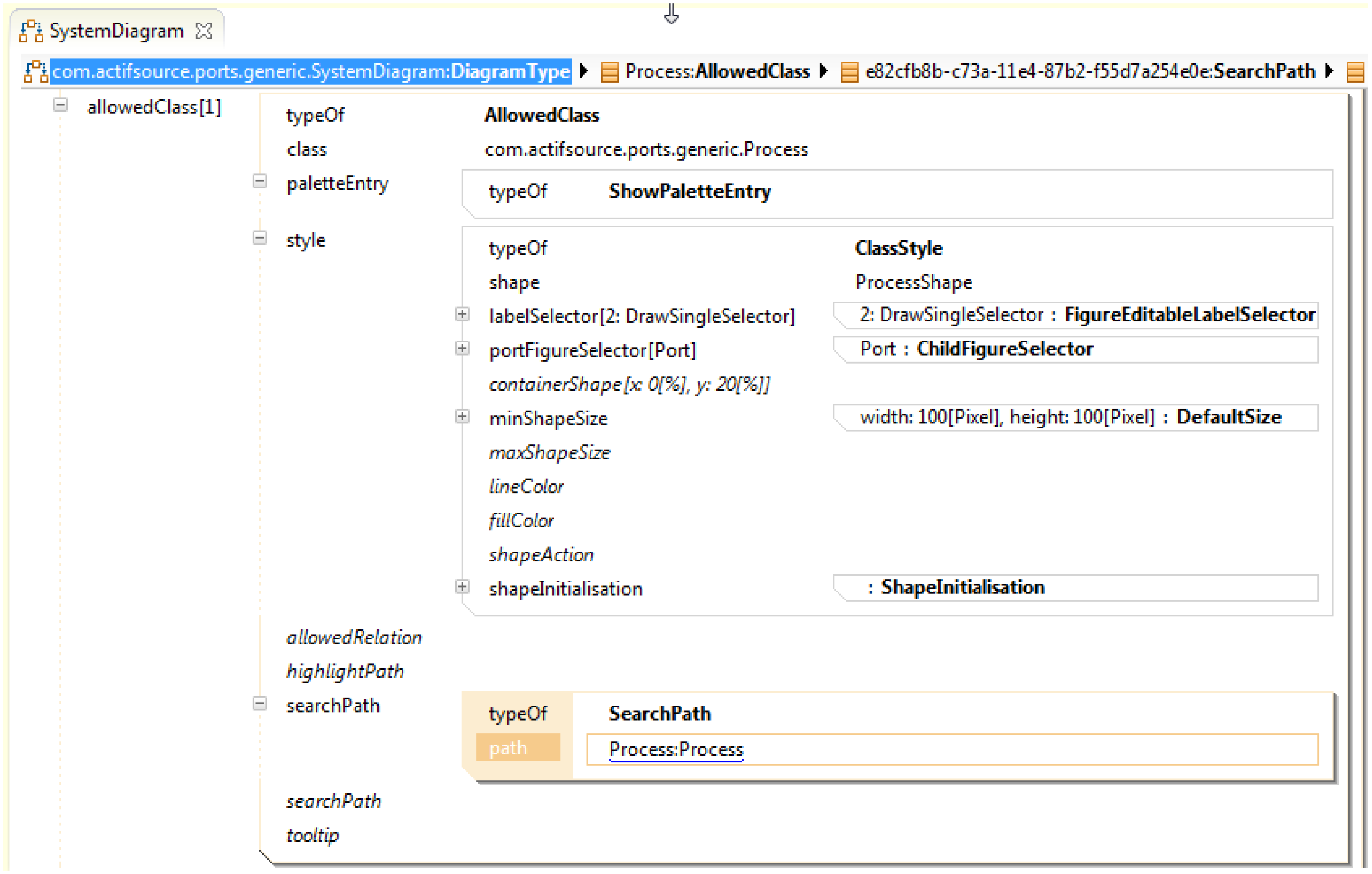


- Open the Domain Diagram SystemA in the **Diagram Editor**
- Select New Note from the Palette
- Right-click on the shape ProcessB to place the note



- Select Edit from the Palette
- Right-click on the note and edit the text
- Write a note (such as *This shape represents ProcessB*)

Part V: Add search functions to Domain Diagrams


SystemDiagram

com.actifsource.ports.generic.SystemDiagram:DiagramType ▸ Process:AllowedClass ▸ e82cfb8b-c73a-11e4-87b2-f55d7a254e0e:SearchPath ▸

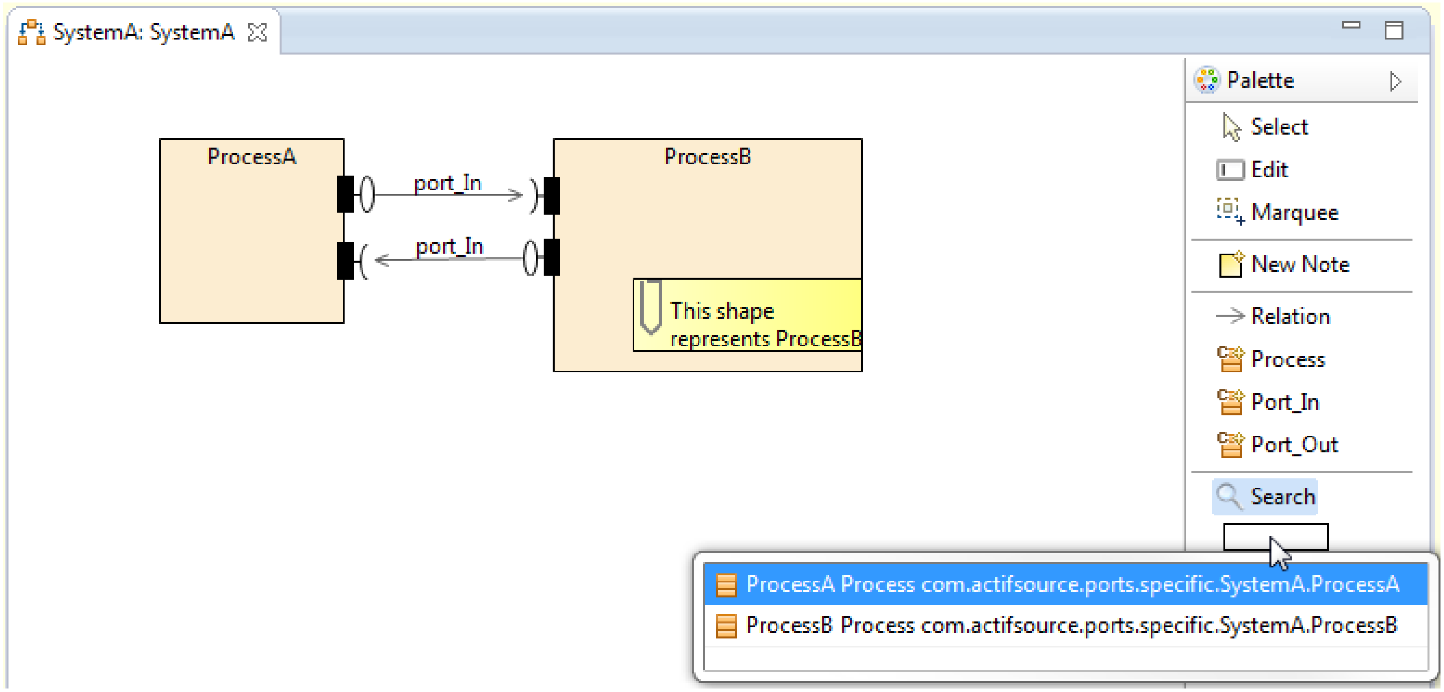
allowedClass[1]

typeOf	**AllowedClass**
class	com.actifsource.ports.generic.Process
paletteEntry	typeOf **ShowPaletteEntry**
style	typeOf **ClassStyle**
	shape: ProcessShape
	labelSelector[2: DrawSingleSelector]: 2: DrawSingleSelector : **FigureEditableLabelSelector**
	portFigureSelector[Port]: Port : **ChildFigureSelector**
	containerShape[x: 0[%], y: 20[%]]
	minShapeSize: width: 100[Pixel], height: 100[Pixel] : **DefaultSize**
	maxShapeSize
	lineColor
	fillColor
	shapeAction
	shapeInitialisation: : **ShapeInitialisation**
allowedRelation	
highlightPath	
searchPath	typeOf **SearchPath**; path Process:Process
searchPath	
tooltip	

We add a search function to our Domain Diagrams that allows us to search for all Processes:

- Open SystemDiagram in the **Resource Editor**
- Create a **SearchPath** in the **AllowedClass** for class Process and define the selector Process:Process as path

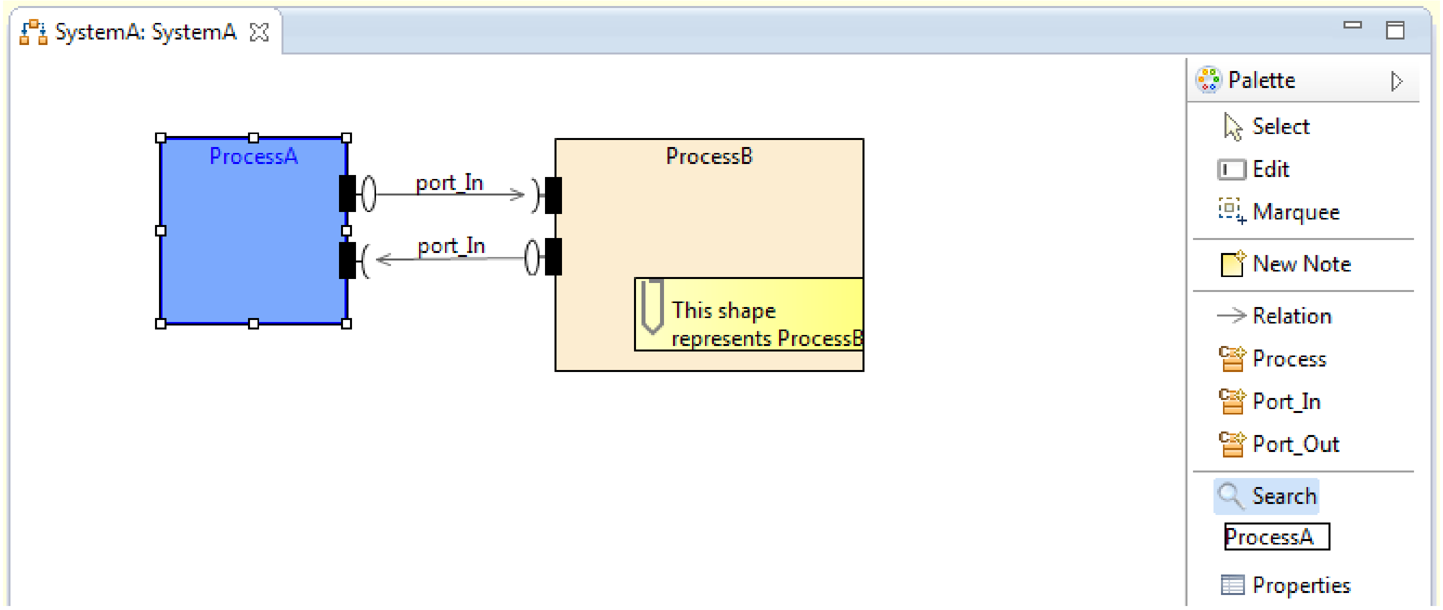
Add search functions to Domain Diagram



ProcessA Process com.actifsource.ports.specific.SystemA.ProcessA
ProcessB Process com.actifsource.ports.specific.SystemA.ProcessB

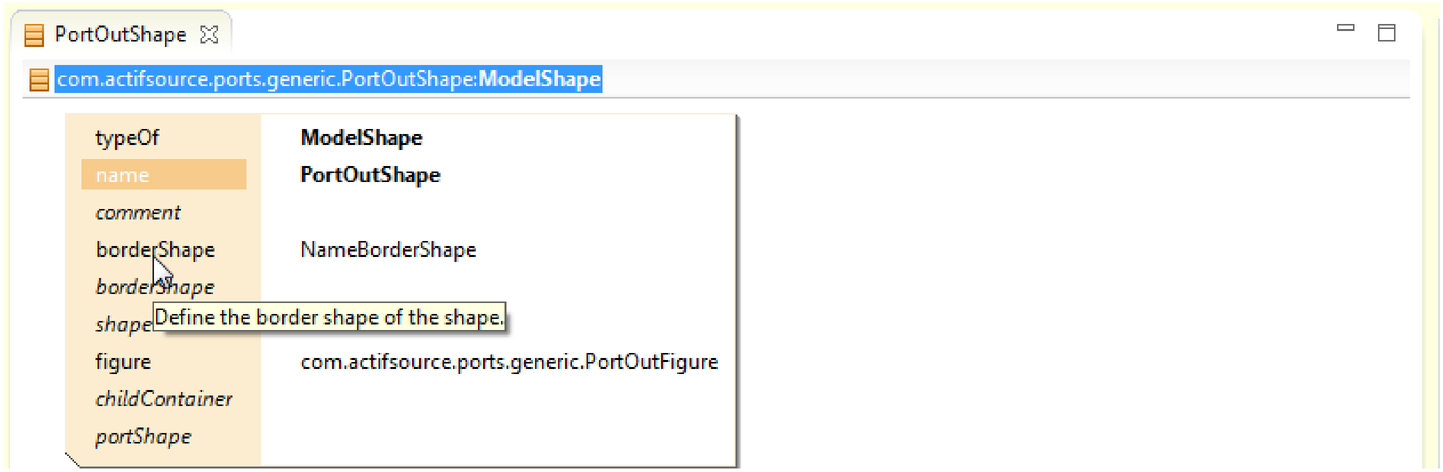
- Close and re-open the Domain Diagram SystemA in the **Diagram Editor**
- Open the Content Assist in the search field and select ProcessA from the proposals

Add search functions to Domain Diagram



Note that the process selected in the search function has been colored (blue).

Part VI: Add labels to ports



PortOutShape

com.actifsource.ports.generic.PortOutShape:ModelShape

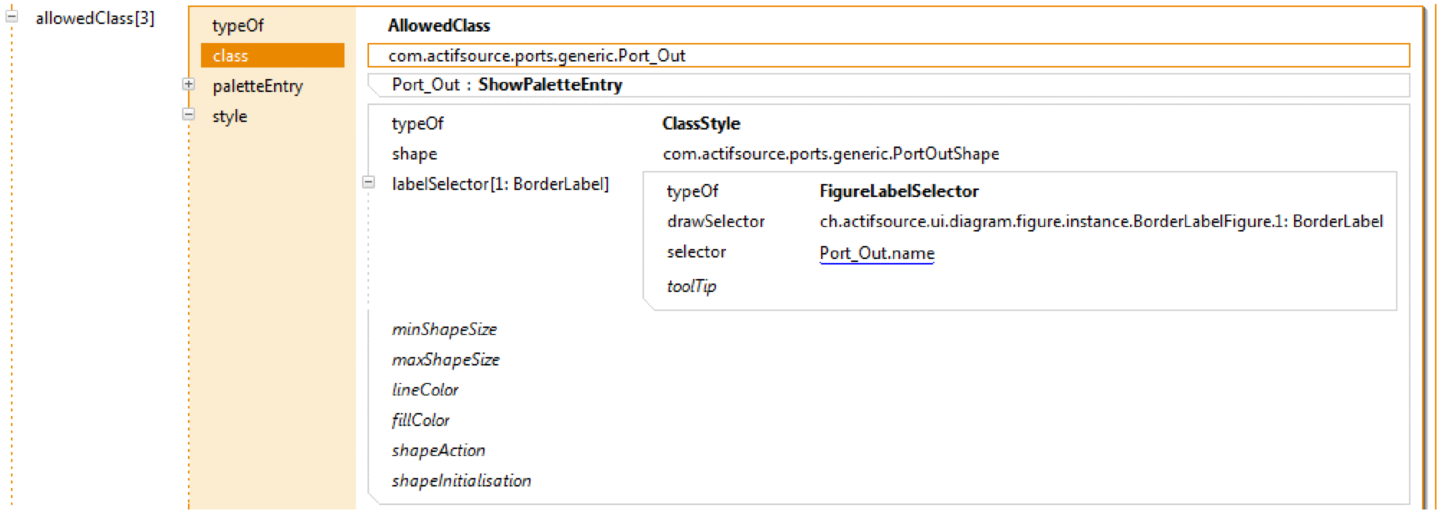
typeOf	**ModelShape**
name	**PortOutShape**
comment	
borderShape	NameBorderShape
borderShape	
shape	
figure	com.actifsource.ports.generic.PortOutFigure
childContainer	
portShape	

Define the border shape of the shape.

Finally, we want to display the name of a port as label in the **Domain Diagram Editor**:

- Open PortOutShape in the **Resource Editor**
- Use the Content Assist to add a **borderShape** statement that refers to **NameBorderShape**

Add labels to ports



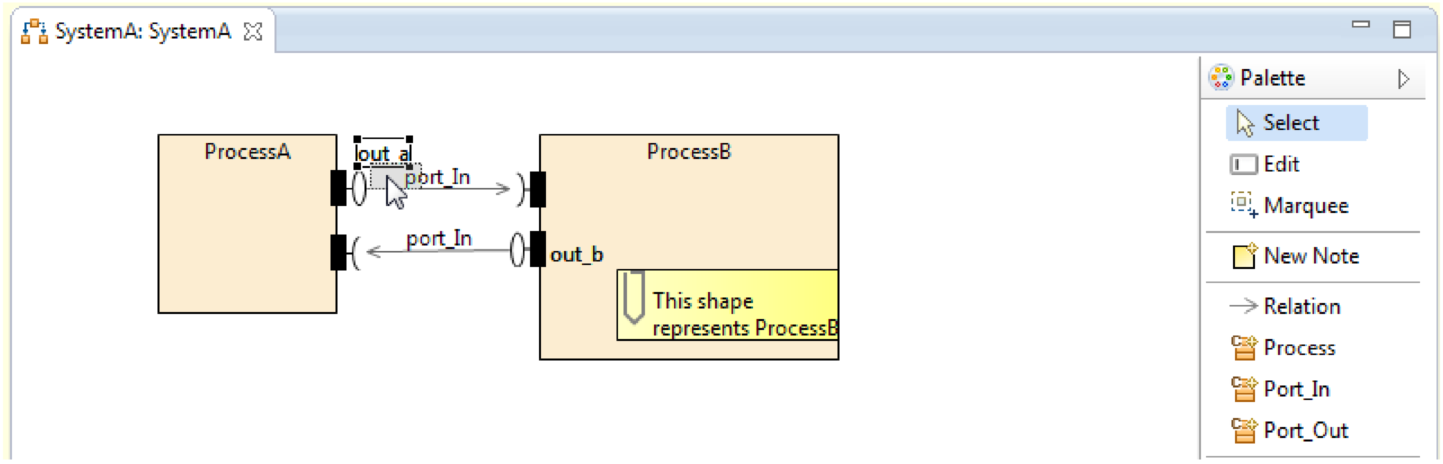
allowedClass[3]

typeOf	**AllowedClass**
class	com.actifsource.ports.generic.Port_Out
paletteEntry	Port_Out : **ShowPaletteEntry**
style	

typeOf	**ClassStyle**		
shape	com.actifsource.ports.generic.PortOutShape		
labelSelector[1: BorderLabel]	typeOf	**FigureLabelSelector**	
	drawSelector	ch.actifsource.ui.diagram.figure.instance.BorderLabelFigure.1: BorderLabel	
	selector	Port_Out.name	
	toolTip		
minShapeSize			
maxShapeSize			
lineColor			
fillColor			
shapeAction			
shapeInitialisation			

- Open SystemDiagram in the **Resource Editor**
- Add a **FigureLabelSelector** with the selector Port_Out.name as Decorator to **labelSelector**

Add labels to ports

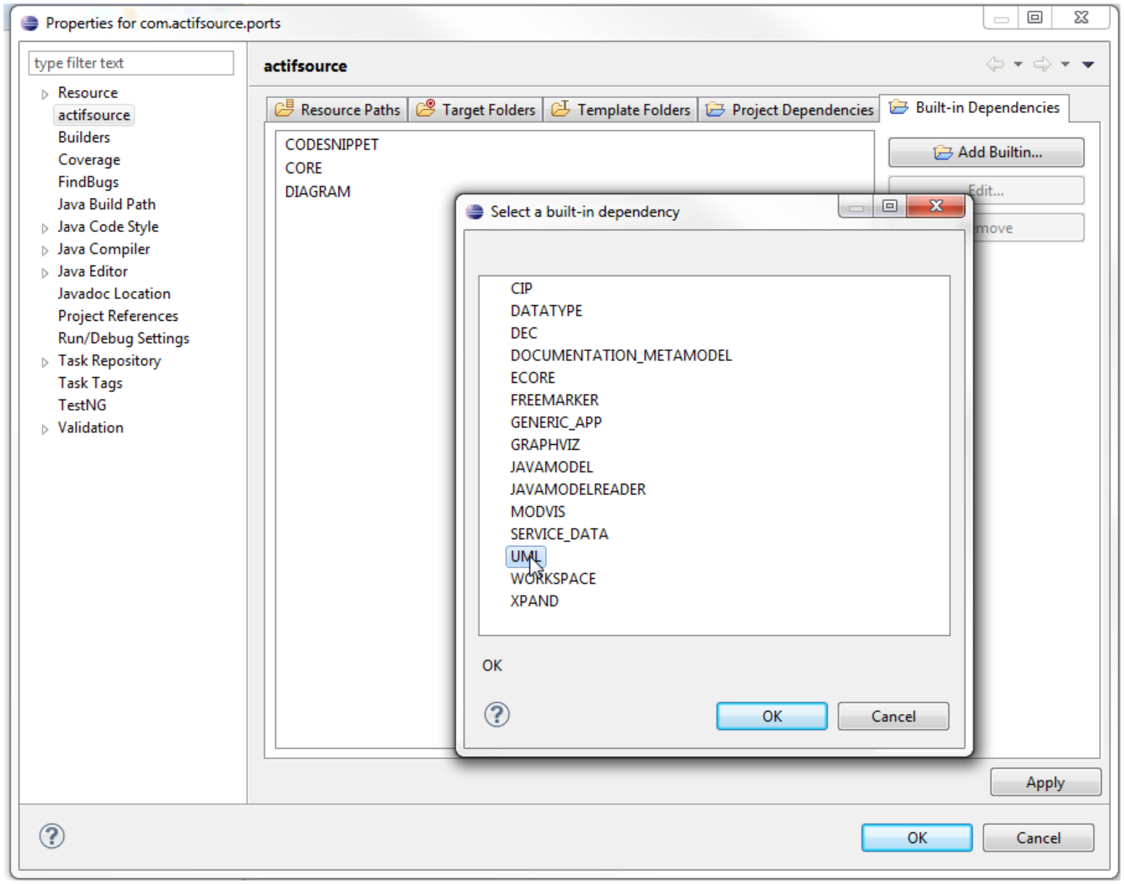


- Open SystemA in the **Diagram Editor**
- Check that the name of outgoing ports are now displayed as labels next to their ports
- Try to move and re-position the labels (in the Select Mode)

Part VII:
Create links to UML State Machines

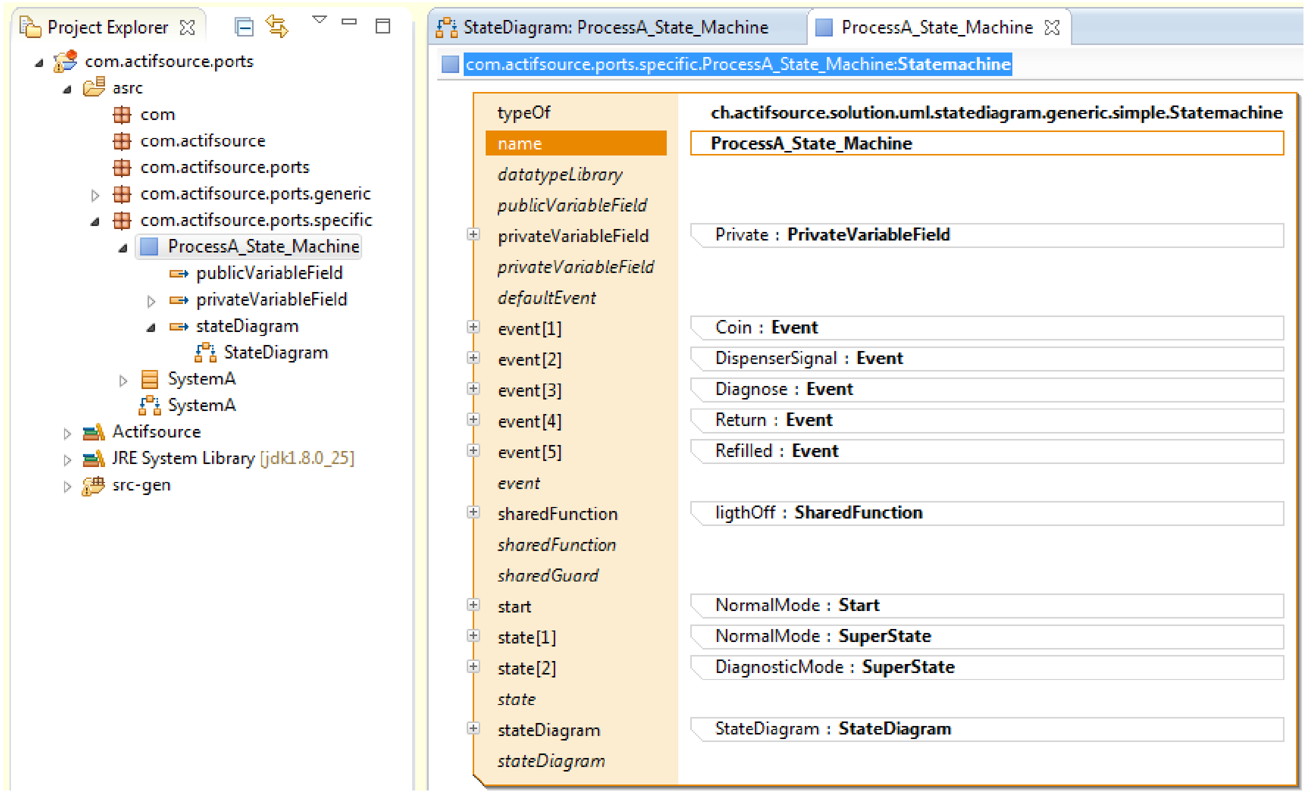
- We want to specify the behavior of our coummicating processes: <u>ProcessA</u> should be a simple coin machine that is connected to a dispenser (<u>ProcessB</u>)
- We create an UML State Machine that defines the behavior of <u>ProcessA</u> and link the state machine to our Domain Diagram

Create links to UML State Machines



First, we make all (built-in) resources needed to build UML diagrams available in our project:

- Right-click on the project com.actifsource.ports and choose Properties from the menu
- Select actifsource in the opened Properties dialog and click on Add Builtin
- Select **UML** from the opened built-in dependency Selection dialog and close both dialogs by clicking OK


Project Explorer
com.actifsource.ports
asrc
com
com.actifsource
com.actifsource.ports
com.actifsource.ports.generic
com.actifsource.ports.specific
ProcessA_State_Machine
publicVariableField
privateVariableField
stateDiagram
StateDiagram
SystemA
SystemA
Actifsource
JRE System Library [jdk1.8.0_25]
src-gen

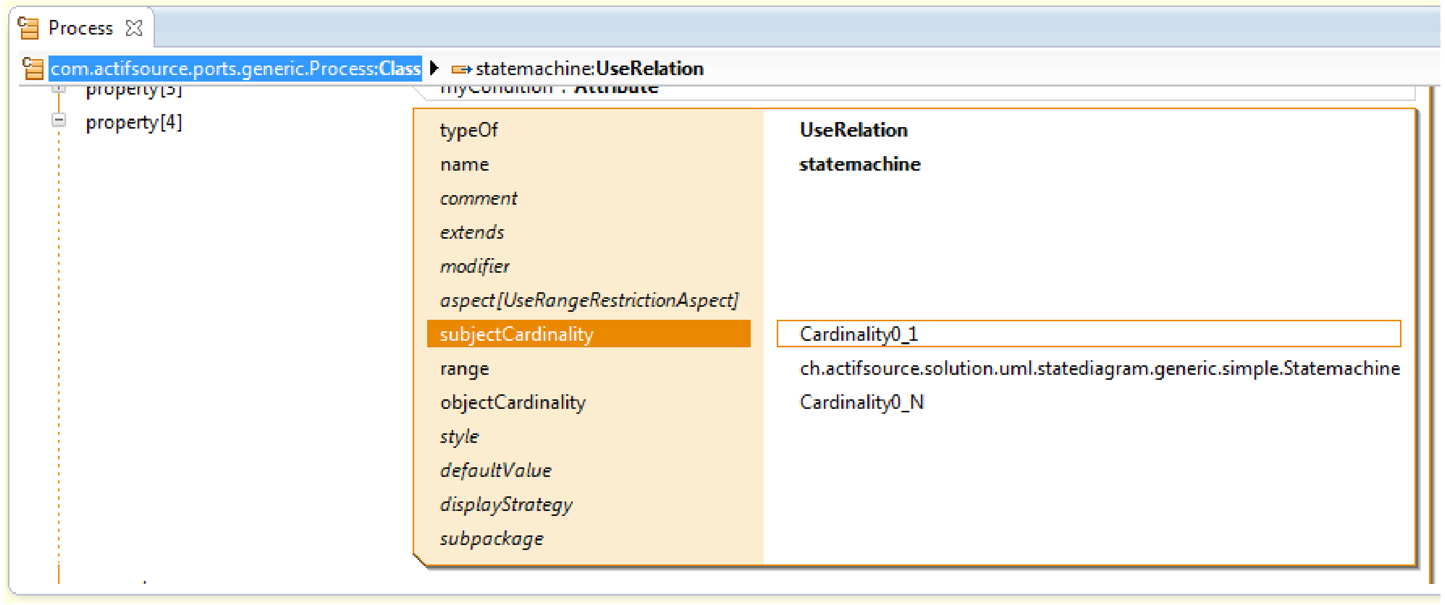
StateDiagram: ProcessA_State_Machine | ProcessA_State_Machine

com.actifsource.ports.specific.ProcessA_State_Machine:**Statemachine**

typeOf	**ch.actifsource.solution.uml.statediagram.generic.simple.Statemachine**
name	**ProcessA_State_Machine**
datatypeLibrary	
publicVariableField	
privateVariableField	Private : **PrivateVariableField**
privateVariableField	
defaultEvent	
event[1]	Coin : **Event**
event[2]	DispenserSignal : **Event**
event[3]	Diagnose : **Event**
event[4]	Return : **Event**
event[5]	Refilled : **Event**
event	
sharedFunction	ligthOff : **SharedFunction**
sharedFunction	
sharedGuard	
start	NormalMode : **Start**
state[1]	NormalMode : **SuperState**
state[2]	DiagnosticMode : **SuperState**
state	
stateDiagram	StateDiagram : **StateDiagram**
stateDiagram	

We create a ch.actifsource.solution.uml.statediagram.generic.simple.Statemachine called ProcessA_State_Machine as described in Actifsource Tutorial – UML State Machines in the package com.actifsource.ports.specific.

Create links to UML State Machines

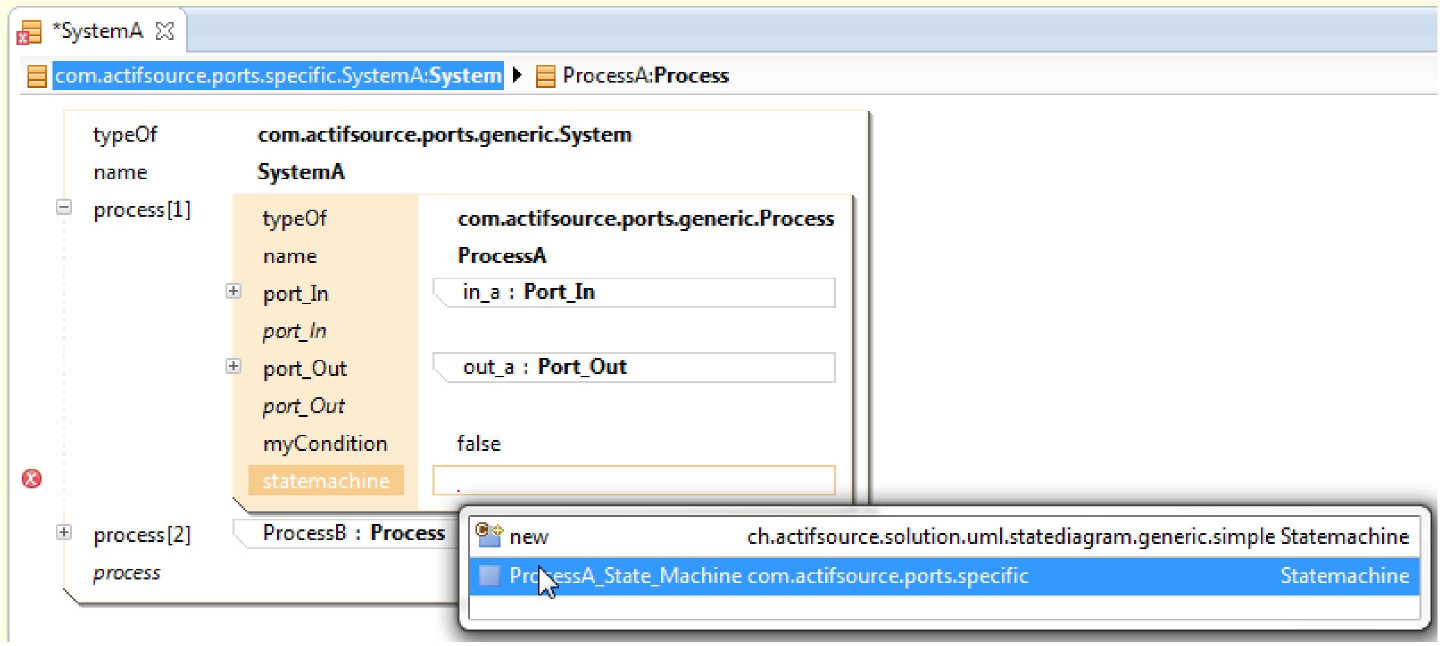


typeOf	**UseRelation**
name	**statemachine**
comment	
extends	
modifier	
aspect[UseRangeRestrictionAspect]	
subjectCardinality	Cardinality0_1
range	ch.actifsource.solution.uml.statediagram.generic.simple.Statemachine
objectCardinality	Cardinality0_N
style	
defaultValue	
displayStrategy	
subpackage	

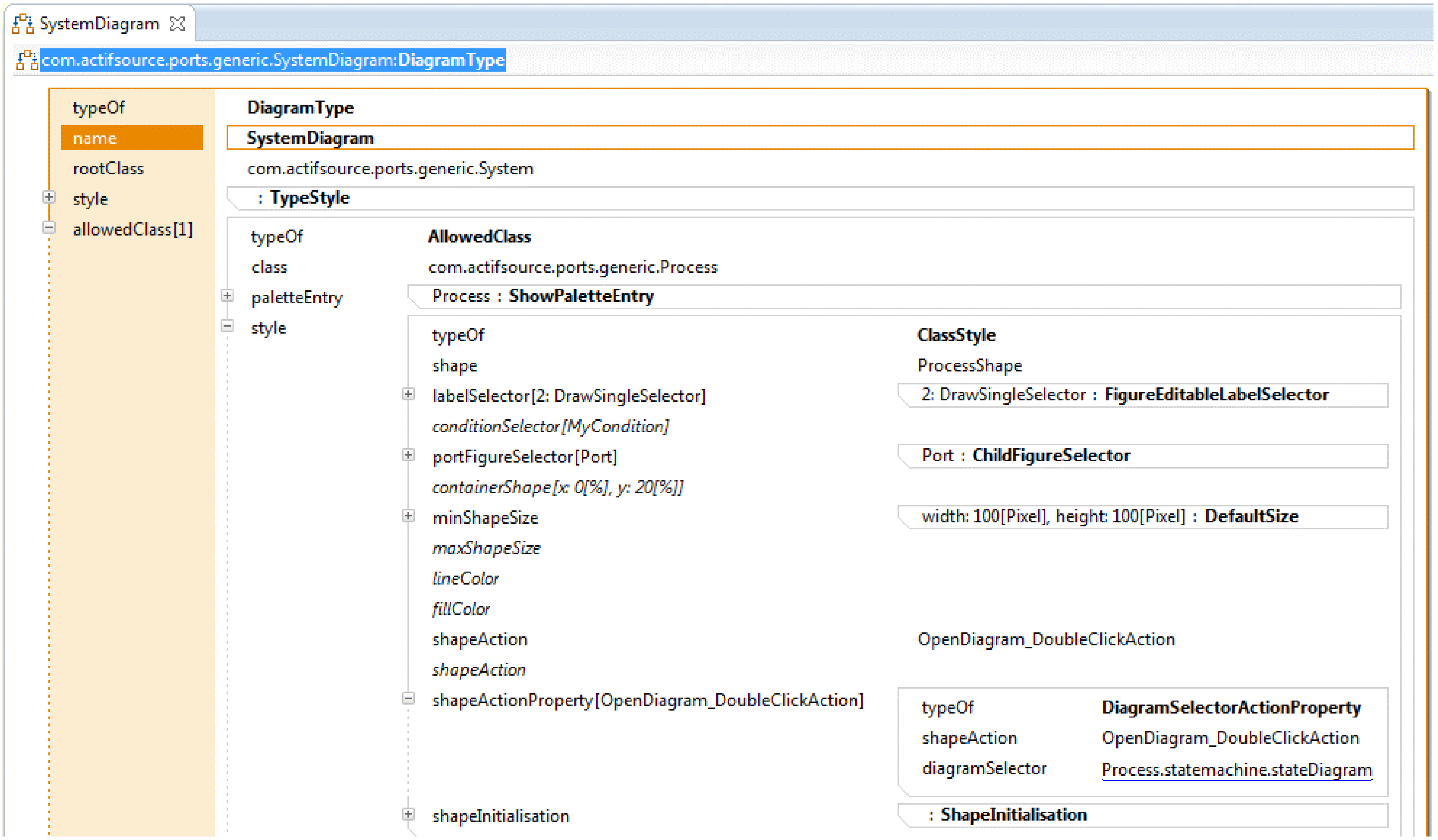
We add a reference to an UML state machine (that describes the behavior of the Process) to Process:

- Open Process in the **Resource Editor**
- Add a UseRelation called statemachine to Process
- Define the range of the relation as ch.actifsource.soltuion.uml.statediagram.generic.simple.Statemachine

Create links to UML State Machines


*SystemA
com.actifsource.ports.specific.SystemA:System
ProcessA:Process
typeOf
com.actifsource.ports.generic.System
name
SystemA
process[1]
typeOf
com.actifsource.ports.generic.Process
name
ProcessA
port_In
in_a : Port_In
port_In
port_Out
out_a : Port_Out
port_Out
myCondition
false
statemachine
process[2]
ProcessB : Process
process
new
ch.actifsource.solution.uml.statediagram.generic.simple Statemachine
ProcessA_State_Machine com.actifsource.ports.specific
Statemachine

- Open SystemA in the **Resource Editor**
- Create a statemachine statement that referers to ProcessA_State_Machine to ProcessA

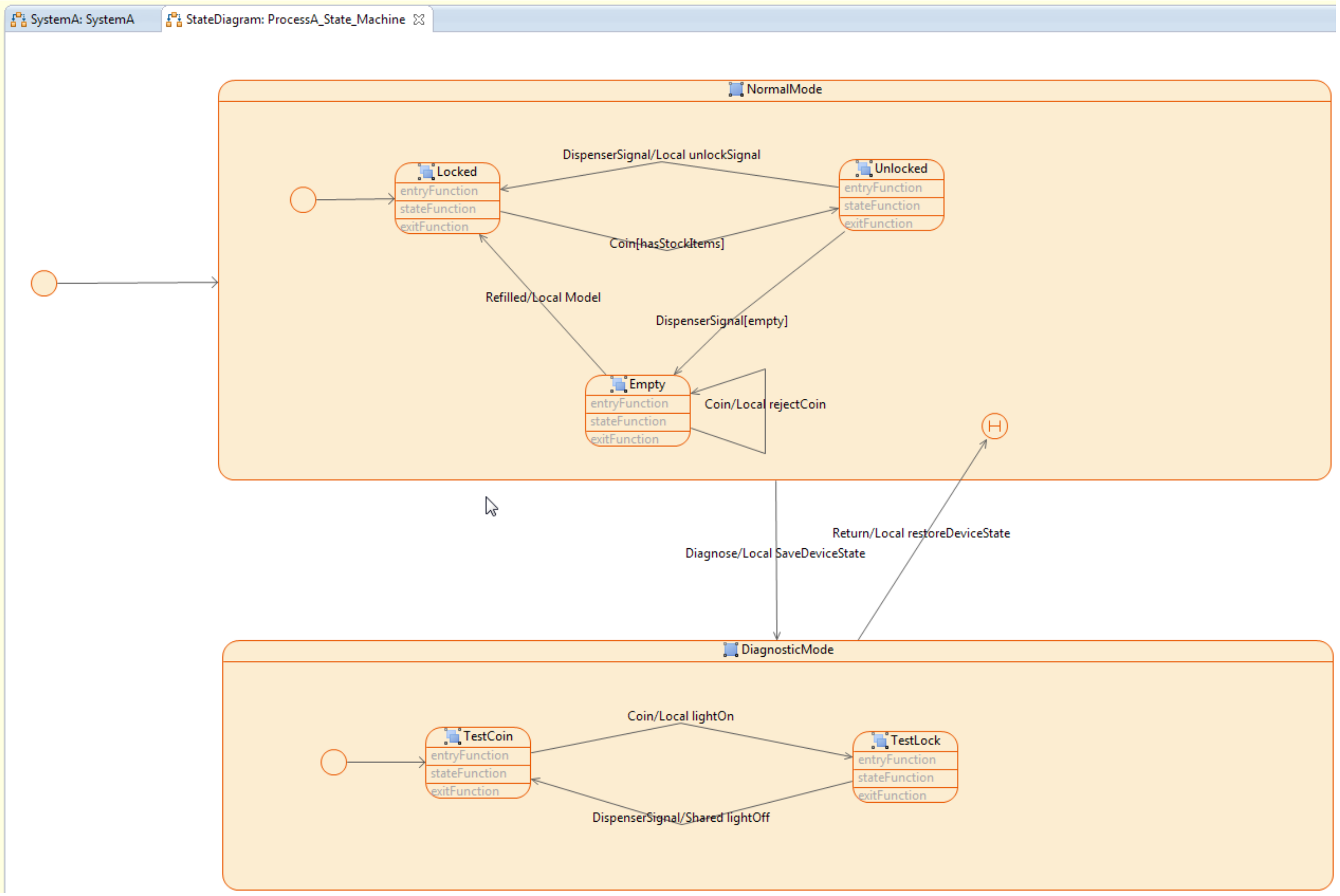

```
SystemDiagram
com.actifsource.ports.generic.SystemDiagram:DiagramType

typeOf            DiagramType
name              SystemDiagram
rootClass         com.actifsource.ports.generic.System
style             : TypeStyle
allowedClass[1]
    typeOf        AllowedClass
    class         com.actifsource.ports.generic.Process
    paletteEntry  Process : ShowPaletteEntry
    style
        typeOf                         ClassStyle
        shape                          ProcessShape
        labelSelector[2: DrawSingleSelector]   2: DrawSingleSelector : FigureEditableLabelSelector
        conditionSelector[MyCondition]
        portFigureSelector[Port]       Port : ChildFigureSelector
        containerShape[x: 0[%], y: 20[%]]
        minShapeSize                   width: 100[Pixel], height: 100[Pixel] : DefaultSize
        maxShapeSize
        lineColor
        fillColor
        shapeAction                    OpenDiagram_DoubleClickAction
        shapeAction
        shapeActionProperty[OpenDiagram_DoubleClickAction]
                                       typeOf          DiagramSelectorActionProperty
                                       shapeAction     OpenDiagram_DoubleClickAction
                                       diagramSelector Process.statemachine.stateDiagram
        shapeInitialisation            : ShapeInitialisation
```

We define an action that is triggered by double-clicking on a process shape in our Domain Diagram and opens the UML state machine associated with the corresponding Process:

- Open SystemDiagram in the **Resource Editor**
- With the help of the Content Assist, add an OpenDiagram_DoubleClickAction to the **AllowedClass** for the class Process
- Create a **DiagramSelectorActionProperty** as **Decorator** for shapeActionProperty and define the selector Process.statemachine.stateDiagram (i.e., the action should open the selected diagram)

Create links to UML State Machines



- Open the Domain Diagram SystemA in the **Diagram Editor**
- Double-Click on ProcessA and make sure that the diagram ProcessA_State_Machine is automatically opened in the **Diagram Editor**