

IDCM – MODELING GUIDE

(DOCUMENTATION FROM [HTTPS://IDCM.WP.MINES-TELECOM.FR](https://idcm.wp.mines-telecom.fr))

Purpose : define a methodology to create UML models under TopCased 4.3 environment that can be analysed with ICDM

Steps:

1. create a **project (main package)**
2. create component **interfaces (main packages)**
3. create **Receive Operation Event** for each operation
4. create **packages** (1 for each component and its evolution)
5. create **components** (external view) and create **ports**
6. **associate interfaces** to components through **ports**

Complete the description of components (internal view)

7. Atomic Component : associate a **state machine**

or

8. Composite Component : associate of **composite diagram**

Overview of a project

Creation of a UML project under Topcased

1. Create a Topcased project
2. Associate a UML model to the project
3. Create a model with a component diagram

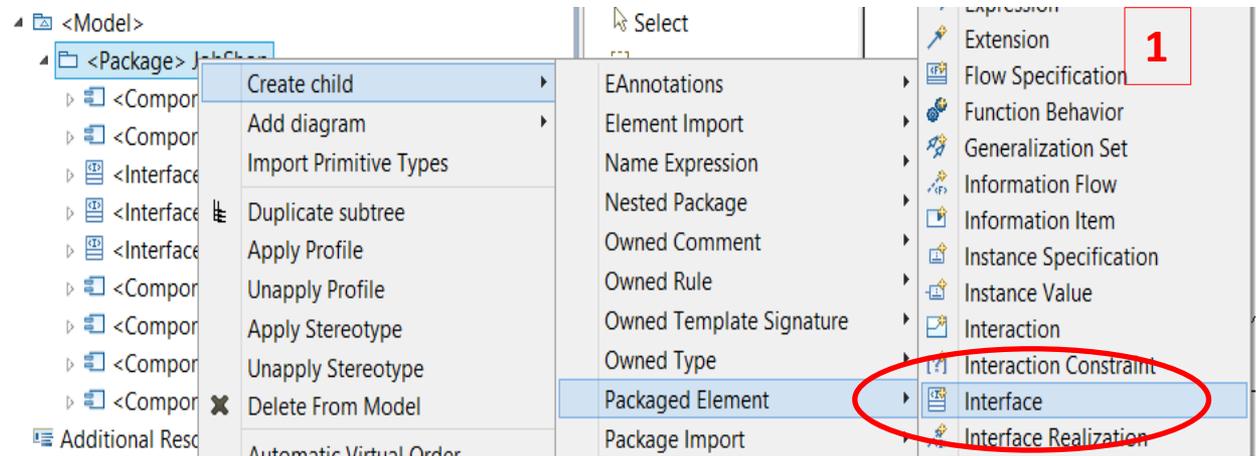
The image illustrates the three steps of creating a UML project in Topcased:

- Step 1:** The 'New Project' wizard is shown. The 'Wizards' list includes 'Papyrus', 'Plug-in Development', 'SVN', 'Topcased', 'TP Template To Project', 'Topcased Project', and 'Topcased Scripting'. The 'Topcased Project' wizard is selected and circled in red.
- Step 2:** The 'New' context menu is shown. The 'UML Model with TOPCASED' option is selected and circled in red.
- Step 3:** The 'New UML model with TOPCASED' dialog is shown. The 'Diagram' dropdown is set to 'Component Diagram' and circled in red. The 'UML Model with TOPCASED' title bar is also circled in red.

Creation of Interfaces

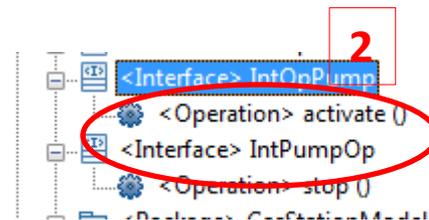
1. Create interfaces

(package or model >> create child >> packaged element >> interface)



2. Add operations to the interfaces

(interface >> create child >> behaviour >> own operation)



Creation of « Receive Operation Events » (used in Trigger definitions)

1. Add receive events

(Package or Model >> create child >> All >> packaged element >> receive operation event)

2. Associate an operation defined in one interface

The screenshot shows the Eclipse Platform interface with the Topcased Modeling environment. The Navigator shows a package structure for 'GasStationModel'. A context menu is open over the package, with 'All' selected, leading to a list of packaged elements. 'Receive Operation Event' is highlighted. The main editor shows three interface definitions: 'IntCustOp', 'IntCustPump', and 'IntOpPump'. A red box labeled '1' is in the top right corner of the editor area.

The Properties view at the bottom right shows the properties of the selected 'Receive Operation Event'. A red box labeled '2' is in the top right corner of the Properties view. The 'UML' property is expanded, and the 'Operation' property is set to '<Operation> activate ()'. The 'Visibility' property is set to 'Public'. A red circle highlights the 'Operation' and 'Visibility' properties.

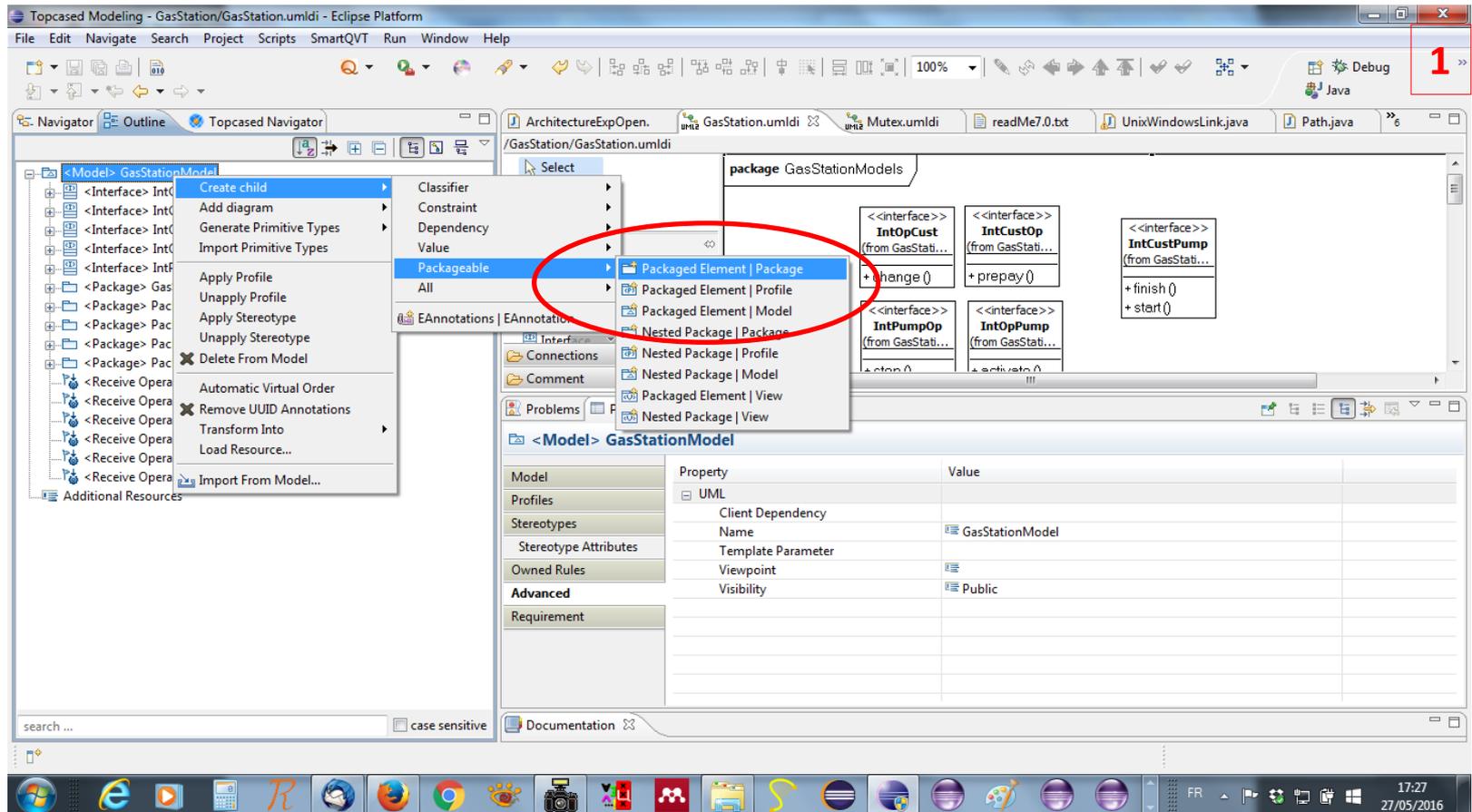
Property	Value
UML	
Client Dependency	
Name	activate
Operation	<Operation> activate ()
Template Parameter	
Visibility	Public

Creation of Packages

(for better oragnisation, use 1 package for 1 component development)

1. Add Package

(Package or Model >> Create child >> Packageable >> Packaged Element | Package)



Creation of Components

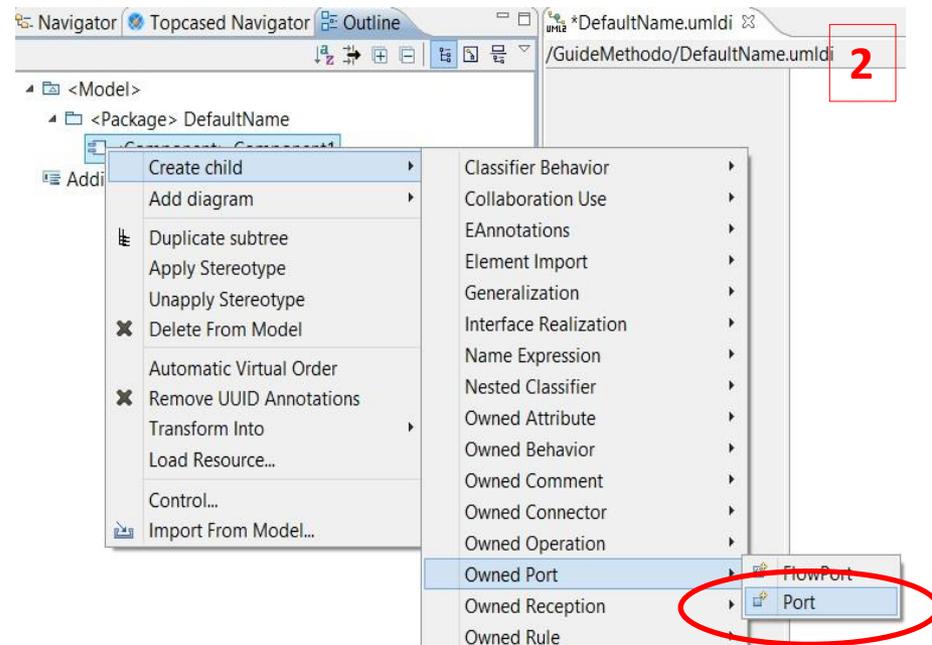
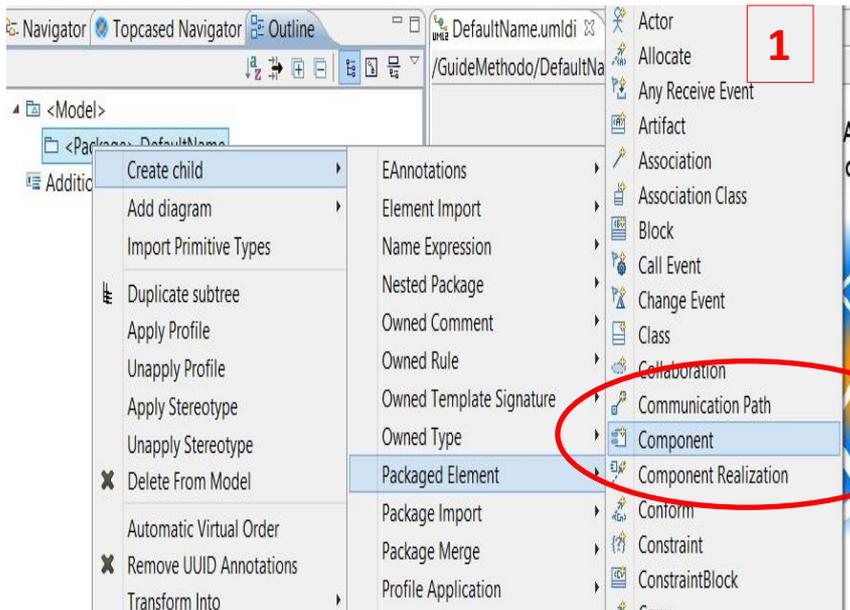
1. Create components

(Package >> Create child >> Packaged Element >> Component)

2. Create ports

(Component >> Create child >> Owned Port >> Port)

Ports are useful for atomic component definition (Trigger and Effect definitions), and for composite component (assembly definitions)



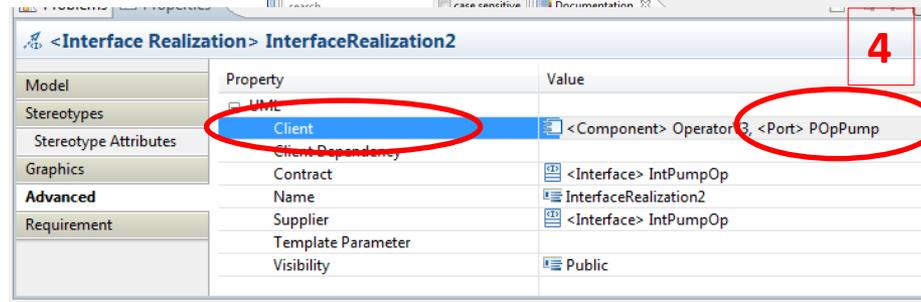
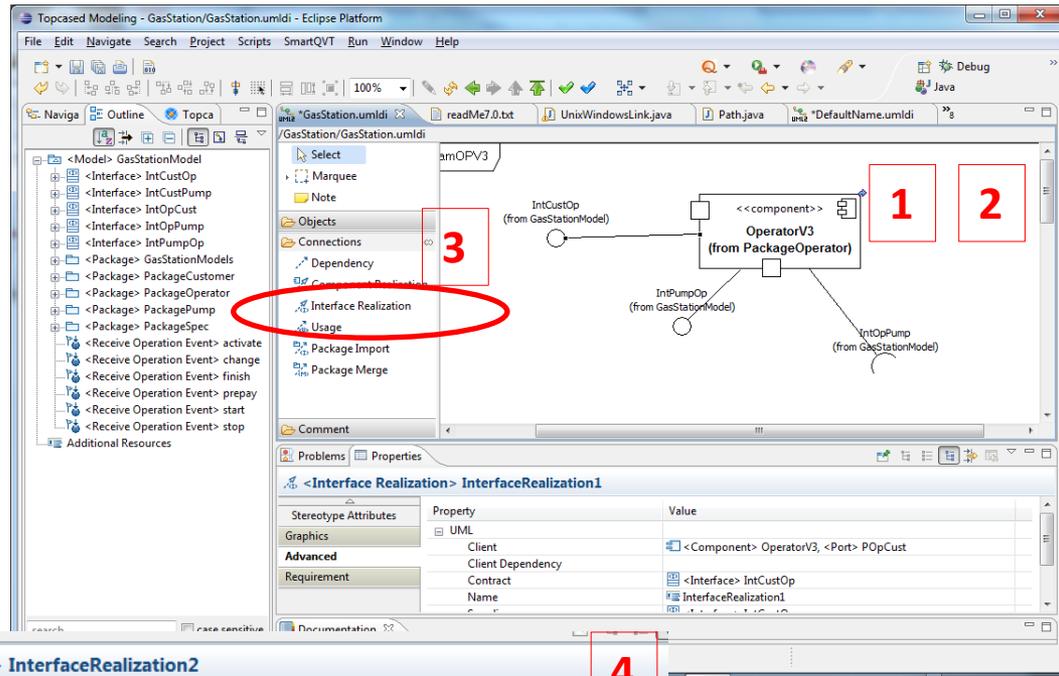
Creation of associations between components and their interfaces (requires the use of a Component Diagram)

1. Create a component diagram –
you may use the default component but better to define at least one component diagram per package
(package >> add diagram >> component diagram)

2. Drag and drop a component and
its interfaces

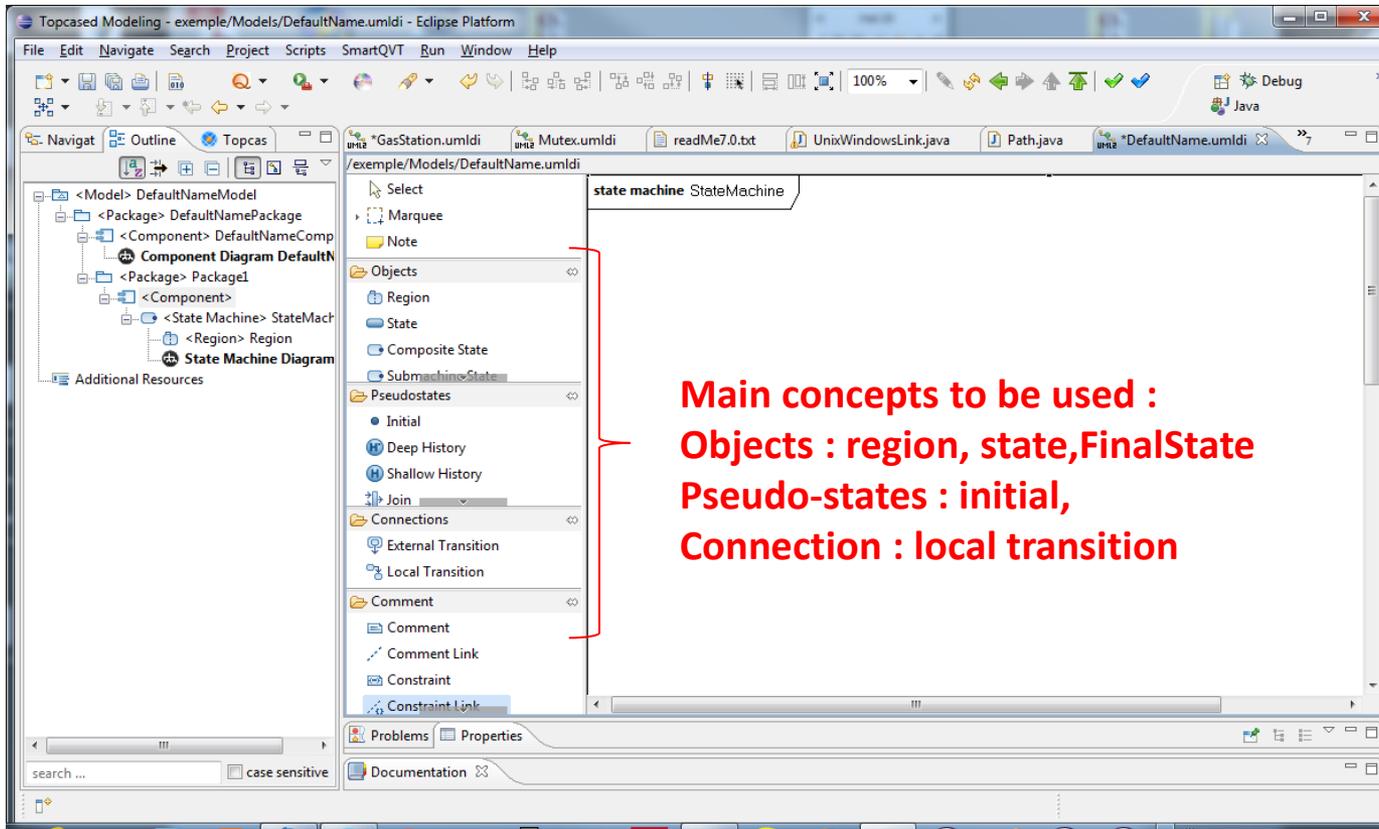
3. Trace connections of type
Interface realization or
Usage
between the component and an interface

4. Associate a port to the connection:
Select the connection
Add a port of the component as a Client



Associate a State Machine to a component

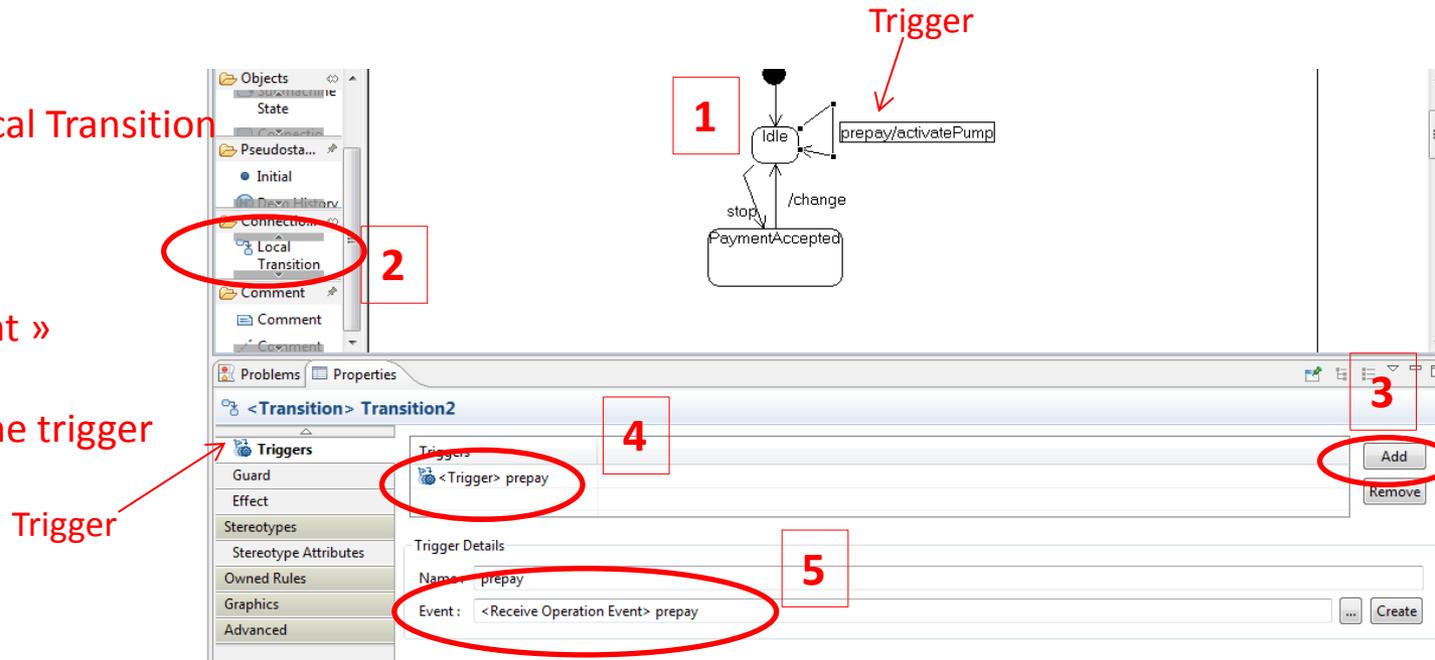
1. Create a state machine
(Component >> add diagram >> state machine diagram)



.../...

Creation of a Transition with a Trigger

1. Create States
2. Create or select a Local Transition
3. Add a trigger
4. Select the Trigger
5. Associate a « Receive Operation Event » created in step 3
6. Associate a port to the trigger



The screenshot shows the 'Properties' window for a trigger named '<Trigger> prepay'. The 'Advanced' section is expanded, showing the 'UML' section with the following properties:

Property	Value
Client Dependency	
Event	<Receive Operation Event> prepay
Name	prepay
Port	<Port> POpCust
Visibility	Public

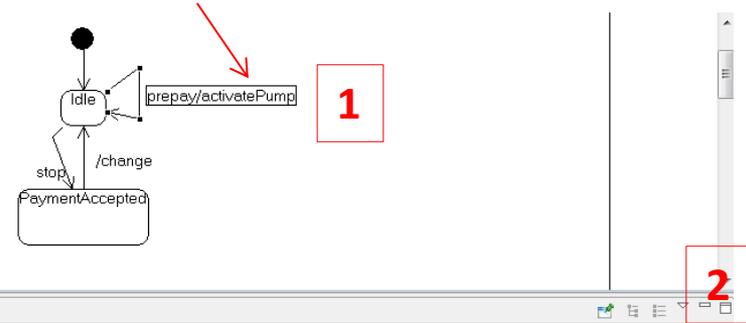
Red boxes and arrows highlight the 'Event' field (step 5) and the 'Port' field (step 6).

.../...

Creation of a Transition with an Effect (requires Activity Diagram creation)

1. Create or select a Local Transition
2. Create an Effect
3. Choose Activity type effect
4. Select the activity and add Activity Diagram
(see [slide 11](#))

Effect



Effect

Properties view for a transition. The 'Effect' tab is selected, and the 'Create...' button is circled in red. A red arrow points to the 'Effect' tab label.

Object selection dialog box. The 'Activity' option is circled in red. A red box with the number '3' is next to the search field.

Context menu for a transition. The 'Add diagram' option is circled in red. A red box with the number '4' is next to the context menu.

Activity Diagram creation

1. Create an Initial Node (Menu Common Menu)
2. Create a Final Node (Menu Common Menu)
3. Create a Call Operation Action (Menu Actions)
4. Associate an Operation (internal or from a required interfaces)
5. Associate a Port if the operations is defined in a required interface
6. Create Connections of type Control Flows (Menu Connections)

The screenshot illustrates the steps for creating an activity diagram. On the left, a tree view shows the structure of the 'activatePump' activity diagram, including nodes like '<Activity Final Node> ActivityFina', '<Call Operation Action> activate', and '<Initial Node> InitialNode1'. The main diagram area shows a sequence: an initial node (1) connected to a call operation action (3) labeled 'activate', which is then connected to a final node (2) via control flows (6). The properties window for the 'activate' call operation action is open, showing the following table:

Property	Value
Client Dependency	
Incoming	<Control Flow> ControlFlow1
In Interruptible Region	
In Partition	
Is Leaf	false
Is Synchronous	true
Name	activate
On Port	<Port> POpPump
Operation	<Operation> activate ()
Outgoing	<Control Flow> ControlFlow2
Redefined Node	
Visibility	Public

.../...

Creation of State with an Activity (required Activity Diagram creation)

state machine StateMachine1

1. Select the State
2. Associate an Activity Diagram
3. Select type doActivity
4. Complete the diagram (explained in [slide 11](#))

Diagram creation

Select the diagram to create :

- Activity Diagram
- Sequence Diagram
- State Machine Diagram

Initialize the diagram with existing model objects

OK Cancel

Behavior type selection

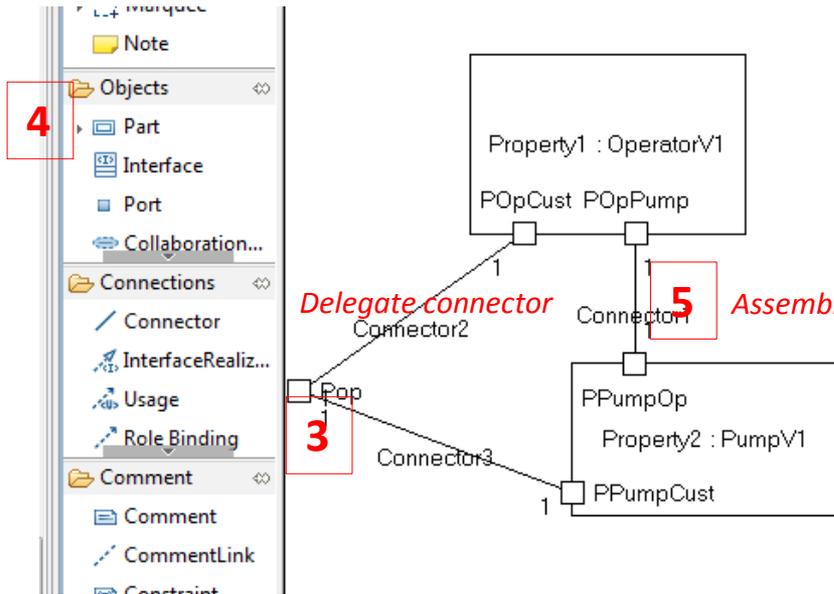
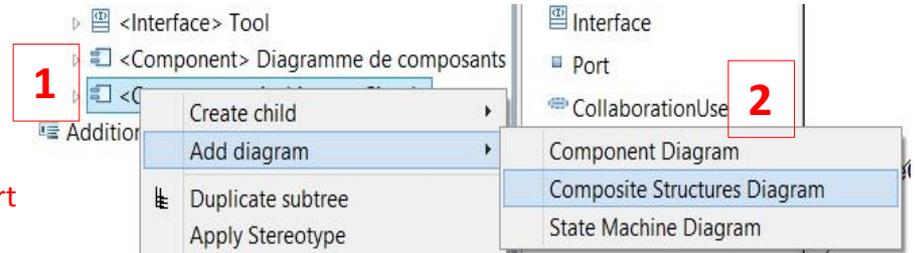
Pick which state behavior you wish to create an Activity Diagram for:

- entry
- doActivity
- exit

OK Cancel

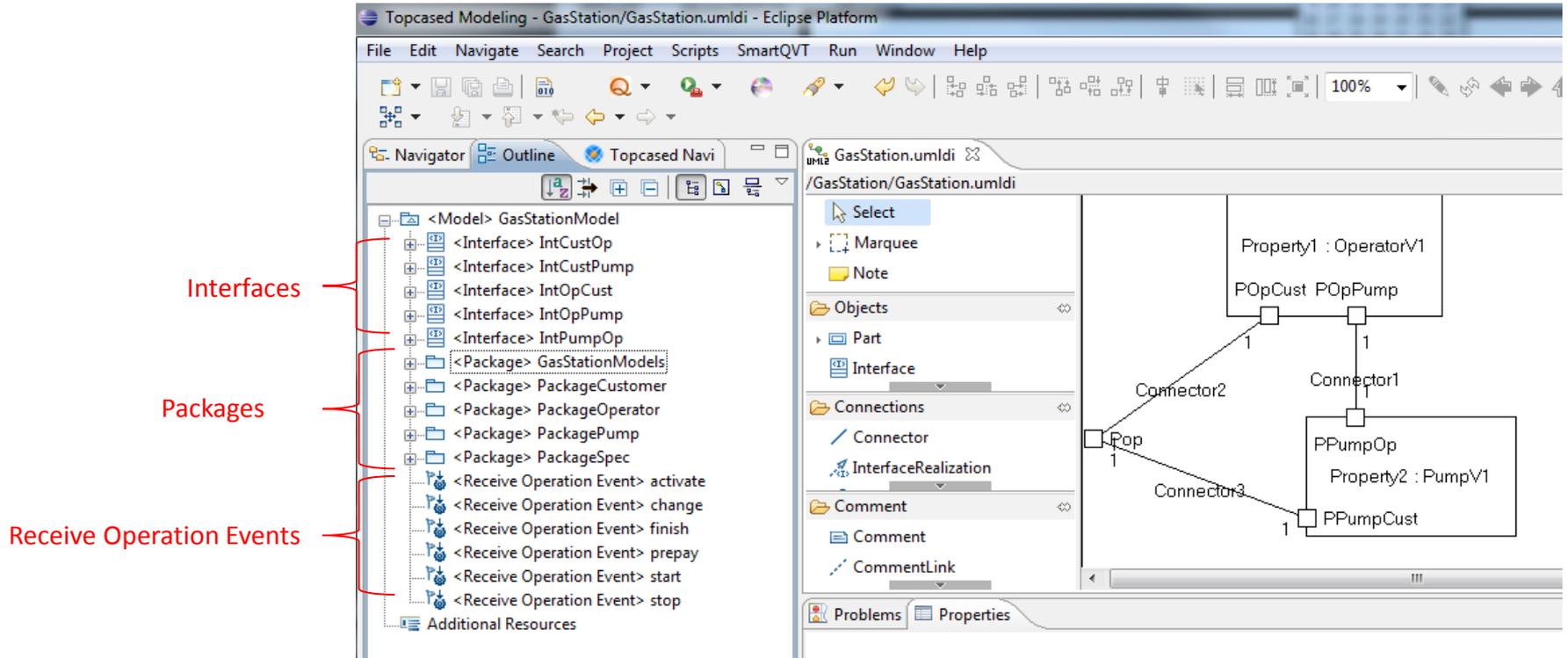
Creation of a Composite Component (requires Composite Structure Diagram creation)

1. Select the Component (see creation slide 6)
2. Associate a Composite Structure Diagram
3. Drag and Drop Ports (see creation slide 6)
4. Select Parts to associate sub-components
5. Connect Ports using Connectors (Menu Connections)
6. Select the type of the Connector
(*assembly* for links of internal Ports, *Delegate* for links with a port of the Composite Component)



<Connector> Connector1		
Model	Property	Value
First End	UML	
Second End	Client Dependency	
Stereotypes	Contract	
Stereotype Attributes	Is Leaf	false
Graphics	Is Static	false
Advanced	Kind	Assembly
Requirement	Name	Connector1
	Redefined Connector	
	Type	
	Visibility	Public

Overview of a Modeling Project



Overview of a Modeling Project

The screenshot displays the Eclipse IDE interface for a UML modeling project. The title bar reads "Topcased Modeling - GasStation/GasStation.uml - Eclipse Platform". The menu bar includes "File", "Edit", "Navigate", "Search", "Project", "Scripts", "SmartQVT", "Run", "Window", and "Help". The toolbar contains various icons for file operations and navigation.

The **Navigator** on the left shows the project structure:

- <Model> GasStationModel
 - <Interface> IntCustOp
 - <Interface> IntCustPump
 - <Interface> IntOpCust
 - <Interface> IntOpPump
 - <Interface> IntPumpOp
 - <Package> GasStationModels
 - <Component> GasStationComponent
 - <Component> gasStationV0
 - <Component> gasStationV1
 - <Connector> Op
 - <Property> C1 : CustomerV1
 - <Property> OP : OperatorV0
 - <Property> P1 : PumpV0
 - <Connector> Pump
 - Composite Structures Diagram gasS**
 - <Component> gasStationV2
 - <Component> gasStationV3_2C_1P
 - <Component> gasStationV4_2C_2P_OPV
 - <Component> gasStationV5_2C_2P_OPV
 - <Component> gasStationV5Star
 - <Component> gasStationV6_2C_2P_OPV

A red bracket on the left side of the Navigator highlights the "Composite Component Package" section, which includes the "Composite Structures Diagram gasS".

The main editor shows the UML diagram for "GasStation.uml". It features three objects: "C1 : CustomerV1" (containing "PCustOp" and "PCustPump"), "P1 : PumpV0" (containing "PPumpCust"), and "OP : OperatorV0" (containing "POpCust"). Relationships are shown as follows:

- A connector labeled "Op" connects "C1 : CustomerV1" (multiplicity 1) to "OP : OperatorV0" (multiplicity 1).
- A connector labeled "Pump" connects "C1 : CustomerV1" (multiplicity 1) to "P1 : PumpV0" (multiplicity 1).

The **Properties** view at the bottom shows the selected diagram "Diagram gasStationV1" with the model name "gasStationV1".

Overview of a Modeling Project

The screenshot displays the Eclipse Platform interface for a modeling project titled "Topcased Modeling - GasStation/GasStation.umldi". The Navigator on the left shows a package hierarchy for "GasStationModel". A red bracket highlights a sub-section labeled "Atomic Component Package", which includes several "State Machine Diagram Operator" components. The Diagram Operator panel on the right shows a state machine diagram with two states: "Idle" and "PaiementAccepted". Transitions are labeled "prepay" and "/change".

Atomic Component Package

```
graph TD; Start(( )) --> Idle[Idle]; Idle -- prepay --> PaiementAccepted[PaiementAccepted]; PaiementAccepted -- /change --> Idle;
```