



# On the Papyrus' USE: Usage, Specialization and Extension

**PhD. Sébastien Gérard**

CEA LIST Senior Expert

Laboratory of model driven engineering  
for embedded systems (LISE)

2010-09-06



[Sebastien.Gerard@cea.fr](mailto:Sebastien.Gerard@cea.fr)



- **Thanks to the CEA Papyrus team for their contributions to this tutorial (following order is not an order... ;-)**
  - Patrick, Tania, Yann, Agnès, Vincent, Saadia, Rémi, Ansgar, Florian and Arnaud.



# Why modeling with standards?

**Standards have traditionally provided major boosts to technological progress !**

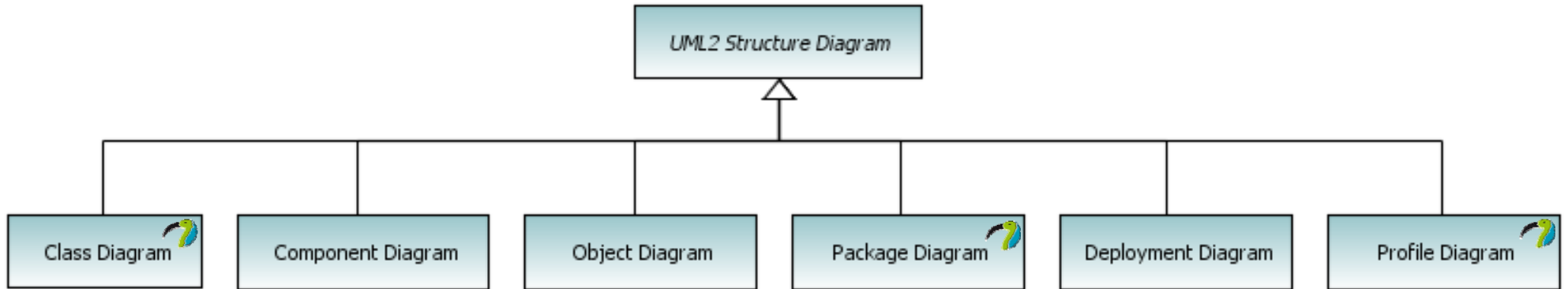
- **But standards enable also vendor independence**
  - Users have a choice of different vendors (no vendor "tie-in")
  - Forces vendors into competing and improving their products
- **The Object Management Group (OMG) has created the Model-Driven Architecture initiative:**
  - A comprehensive set of standards in support of MBE including standard modeling languages: **UML2, MARTE and SysML.**



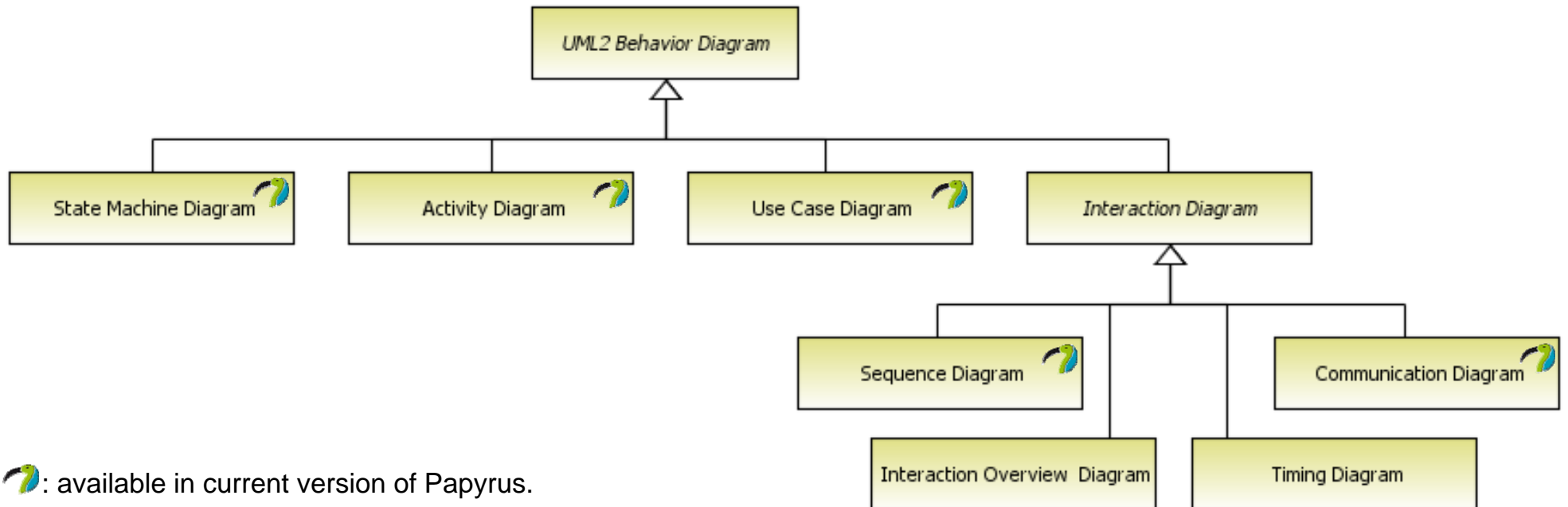


# UML2, a family of modeling languages

- 6 diagram kinds for structure modeling



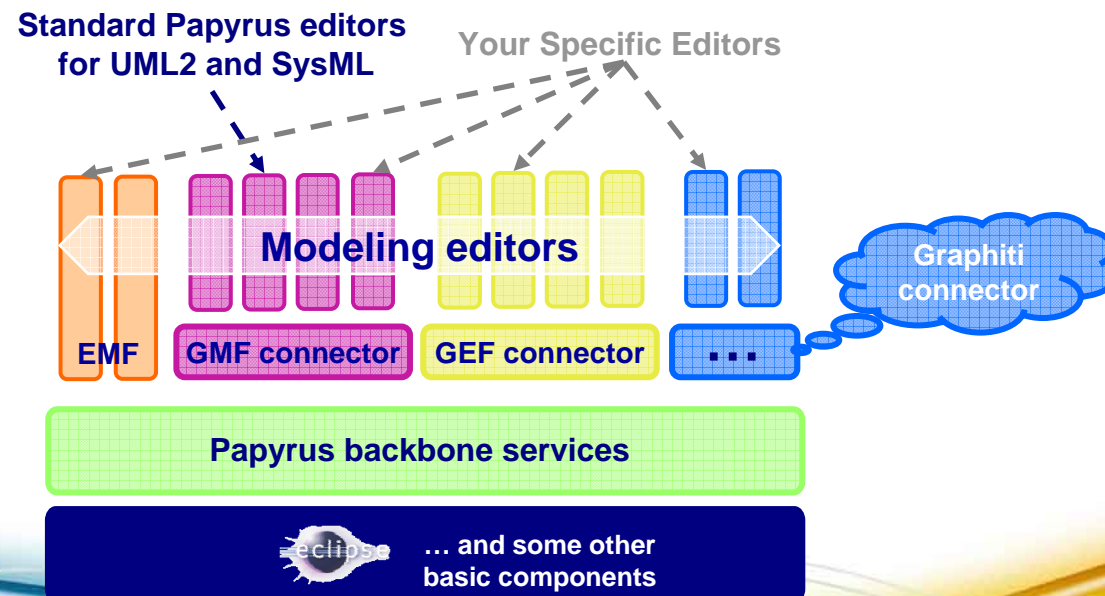
- 7 diagram kinds for behavior modeling



: available in current version of Papyrus.



- **Eclipse based as usual...**
  - Based on well known Eclipse modeling components
    - EMF, GMF, UML2, Modisco, xtext...
    - UML2 and SysML diagram based in GMF (custom generation)
- **... built as an integration platform for diagrams**
  - Supporting various modeling languages
    - Not necessary UML2
  - Graphical or text-based editors
  - Supporting several frameworks
    - GMF, GEF ready (connector available)
    - Extensible (dedicated connector) to future frameworks (Graphiti)



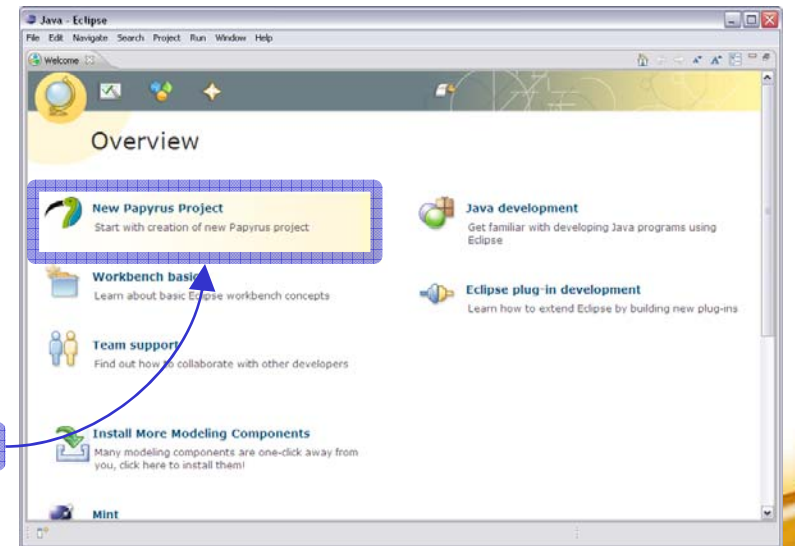
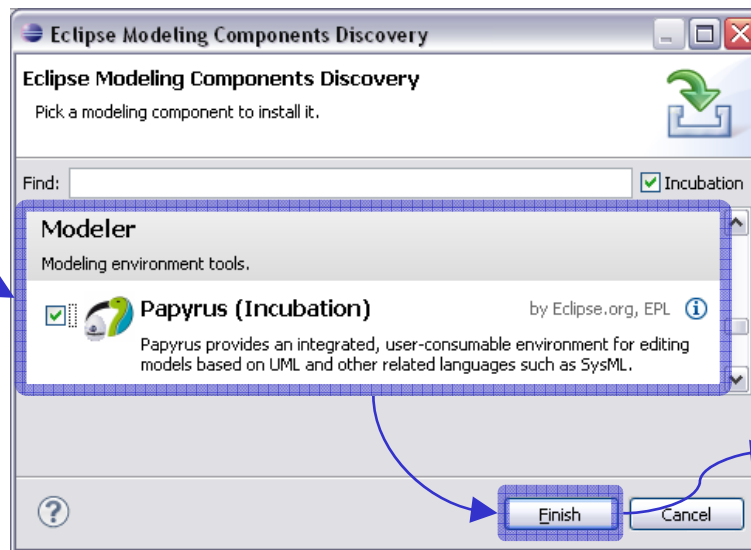
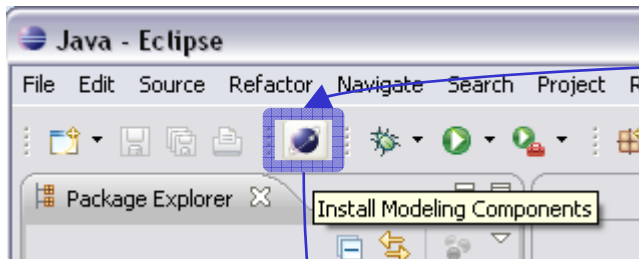


# Papyrus standard download

## • Via the standard Eclipse Modeling Platform

### ▪ Scenario:

- Download the Eclipse Modeling Platform ([www.eclipse.org/downloads](http://www.eclipse.org/downloads)),
- Unzip the downloaded file and start Eclipse.exe,
- Launch the Modeling discovery site update,
- Check Papyrus and start installation.





# Papyrus nightly build version download

## • Via the Papyrus update site

### ▪ Scenario:

- Download the Eclipse Modeling Platform ([www.eclipse.org/downloads](http://www.eclipse.org/downloads)),
- Unzip the downloaded file and start Eclipse.exe,
- In the menu bar, check Help > Install New Software...

The sequence of screenshots illustrates the process of installing Papyrus 0.7.1 Update:

- Help Menu:** The 'Install New Software...' option is selected under the 'Help' menu.
- Install Dialog:** The 'Add...' button is clicked to add a new software site.
- Add Repository Dialog:** The 'Name' is 'Papyrus 0.7.1 Update' and the 'Location' is 'http://download.eclipse.org/modeling/mdt/papyrus/'.
- Install Details Dialog:** A list of software components to be installed is shown, including MDT Papyrus - Activity Diagram editor, MDT Papyrus - Auto-Layout task (Incubation), MDT Papyrus - Backbone, MDT Papyrus - Class Diagram editor, MDT Papyrus - Communication Diagram editor, MDT Papyrus - Composite Diagram editor, MDT Papyrus - Direct editors for UML, and MDT Papyrus - Package Diagram editor.
- Review Licenses Dialog:** The 'I accept the terms of the license agreement' checkbox is checked, and the 'Finish' button is clicked.



**Papyrus - Mozilla Firefox**  
[Fichier](#) [Edition](#) [Affichage](#) [Historique](#) [Marque-pages](#) [Outils](#) ?  
<http://www.eclipse.org> [CIR se](#)

**Papyrus**  
[Home](#) [Downloads](#) [Users](#) [Members](#) [Committers](#) [Resources](#) [Projects](#) [About Us](#) [Google](#) [Custom Search](#)

**Download**  
Eclipse Distribution, Update Site, Droptins

**Support**  
Bug Tracker, Newsgroup, Professional Support

**Documentation**  
Tutorials, Examples, Videos, Online Reference

**Getting Involved**  
CVS, Workspace Setup, Wiki, Committers

**Papyrus**  
 Papyrus is aimed at providing an integrated and user-consumable environment for editing any kind of EMOF model and particularly supporting UML2 and related modeling languages such as SysML and MARTE. Papyrus provides diagram editors for EMOF-based modeling languages amongst them UML2 and SysML and the glue required for integrating these editors (EMF-based or not) with other MBO and MBO2 tools.

Papyrus also offers a very advanced support of UML2 profiles that enables users to define editors for DSLs based on the UML2 standard. The main feature of Papyrus regarding this latter part is a set of very powerful customization mechanisms which can be leveraged to create user-defined Papyrus perspectives and give it the same look and feel as a "pure" DSL editor.

**Current Status**  
 Papyrus team has released the first release 0.7.0 of Papyrus; you may download it here or using the Modeling Eclipse Package, using the discovery interface.

**Headlines in the web**  
 Papyrus presentation at the EclipseCon 2010 (about 5 months ago by Papyrus team)  
 Rami Schneidmayer, Raphael Fiebus and Karel Husky presented Papyrus MDT project on a talk at the EclipseCon 2010. The presentation lasted 25 minutes and mainly focused on the goals of this project.

**UML2**  
 Papyrus is graphical editing tool for UML2 as defined by OMG. Papyrus targets to implement 100% of the OMG specification.

**DSL**  
 Papyrus provides a very advanced support for UML2 profiles enabling support for "pure" DSL. Every part of Papyrus may be customized: model explorer, diagram editors, property editors, etc.

**SysML**  
 Papyrus provides also a complete support for SysML, in order to enable model-based system engineering. It includes an implementation of the SysML static profile and the specific graphical editors required for SysML.

**Text in Papyrus**  
 Papyrus is graphical tool also textual. It is hence possible to edit model elements using contextual text editors enabling syntax highlight, completion and context assist. It is also a customizable feature of Papyrus.

**And much more...**  
 Read the Papyrus user documentation and join the discussion at the forum to understand how powerful Papyrus is. Want to know more? [About Papyrus](#)

[Home](#) [Privacy Policy](#) [Terms of Use](#) [Copyright Agent](#) [Legal](#) [Contact Us](#) [Copyright © 2010 The Eclipse Foundation. All Rights Reserved.](#)

**Papyrus Update Sites - Mozilla Firefox**  
[Fichier](#) [Edition](#) [Affichage](#) [Historique](#) [Marque-pages](#) [Outils](#) ?  
<http://www.eclipse.org/modeling/mdt/papyrus/updates/index.php>

**Papyrus Update Sites**  
[Home](#) [Downloads](#) [Users](#) [Members](#) [Committers](#) [Resources](#) [Projects](#) [About Us](#) [Google](#)

**Papyrus Update Sites**  
 Recommended installation, using Eclipse Modeling Package:

- You can download **Eclipse Modeling Package** for your own platform.
- Use the discovery interface ("Help" => "Install Modeling Component") and select Papyrus.
- Proceed through installation steps.
- Papyrus is now ready to use!

**How to add the Papyrus update site**

**Installation using update sites:**  
 There are several different ways to add a new update site to the list of sites available from the Install Manager. In all cases, the site location (i.e. the Web URL or the archived Update Site provided above) is the only required item.

**Main update site: (Recommended)**

- <http://download.eclipse.org/modeling/mdt/papyrus/updates/releases/> (Eclipse Helios Update)

**Development update sites (version 0.7.1):**

- <http://download.eclipse.org/modeling/mdt/papyrus/updates/nightly/helios> (Eclipse Helios Update)





# Papyrus wellcome page

**Overview**

- New Papyrus Project**  
Start with creation of new Papyrus project
- Workbench basics**  
Learn about basic Eclipse workbench concepts
- Team support**  
Find out how to collaborate with other developers
- Install More Modeling Components**  
Many modeling components are one-click away from you, click here to install them!
- Java development**  
Get familiar with developing Java programs using Eclipse
- Eclipse plug-in development**  
Learn how to extend Eclipse by building new plug-ins

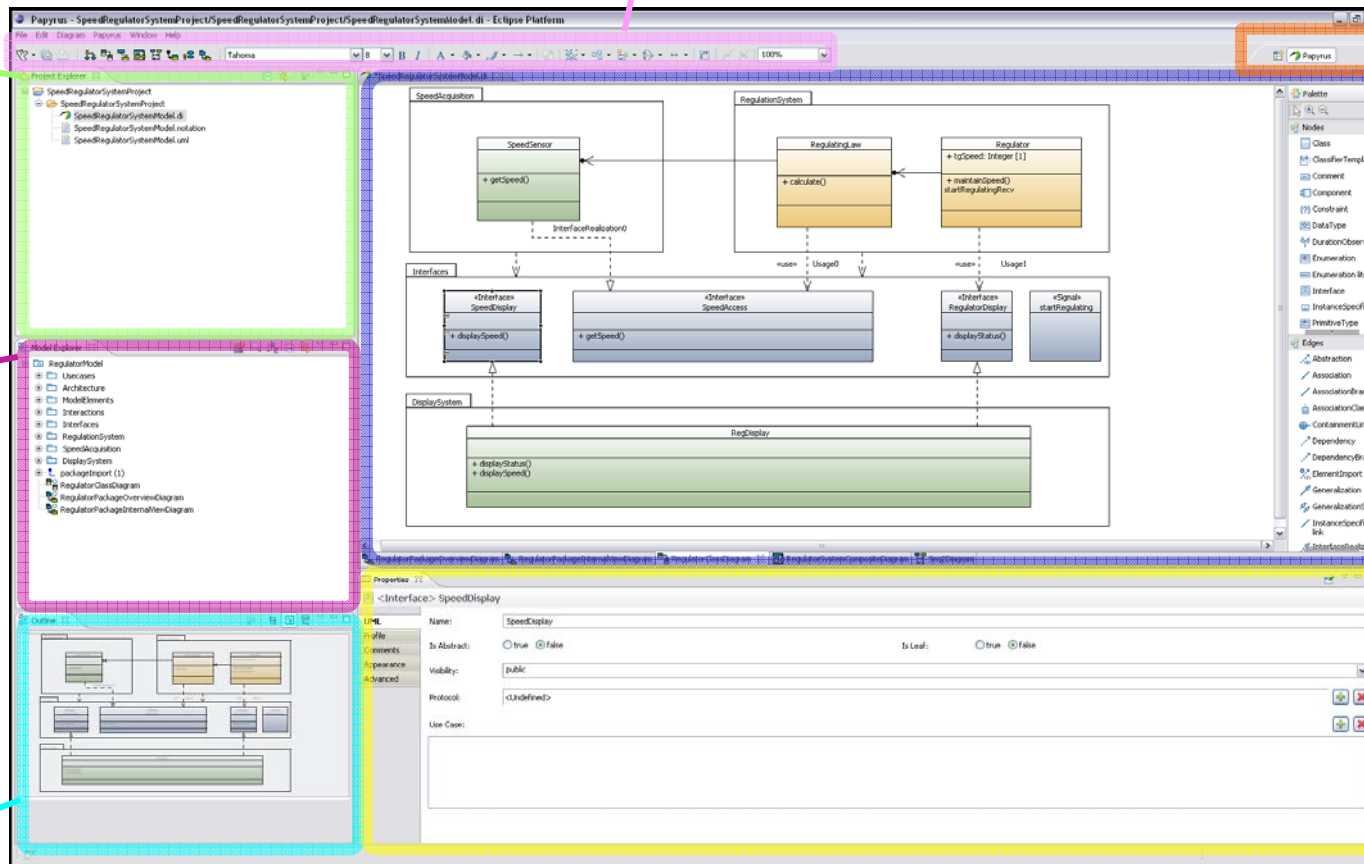


# Outlines of the Papyrus perspective

**Project explorer:** used to manage Papyrus projects at file system level.

**Main toolbar:** diagram creation, graphical editing (align, distribute...), show /hide, ...

**Perspective:** switch the modeling context, define windows (eclipse views) arrangement, define the list of available diagrams, define the available menus and toolbars.



**Model editors:** model editor enabling to edit models through a given modeling language.

**Outline view:** provide overview of the model (read only).

**Model explorer:** tree-based model editor covering the whole model.

**Property view:** form-based model editor enabling to view & edit model element properties.



## • Creating a Papyrus project

- In the Menu bar, click on: File > New>Papyrus Project

**1**

File Edit Window Help

New Alt+Shift+N Papyrus Project

Close Ctrl+W

Close All Ctrl+Shift+W

Save Ctrl+S

**2**

Enter a project name and press Next.

New Papyrus Project

Papyrus Project

Create a New Papyrus Project

Project name: SpeedRegulatingSystemProject

Use default location

Location: D:\HelloWS\runtime-EclipseApplication-2\SpeedRegulatingSystem

Working sets

Add project to working sets

Working sets: [dropdown] Select...

**3**

Select a modeling language (e.g. SysML).

New Papyrus Model

Initialization information

Select language of the diagram

Diagram Language:

UML

Profile

SysML

**3**

3

New Papyrus Model

Initialization information

Select name and kind of the diagram

Diagram Name: MyBlockDefinitionDiagram

Select a Diagram Kind:

SysML Parametric Diagram

SysML Block Definition Diagram

SysML Requirement Diagram

SysML Internal Block Diagram

You can load a template:

Remember current selection

**4**

4

Project Explorer

SpeedRegulatingSystemProject

MyNewProject.di

MyNewProject.notation

MyNewProject.uml

Model Explorer

SysMLmodel

profileApplication (11)

MyBlockDefinitionDiagram

New created UML model:

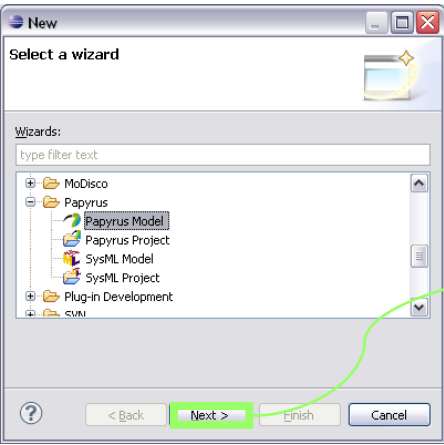
- \*.di: tool metadata
- \*.notation: graphical data
- \*.uml: UML model data



## • Creating a new Papyrus model

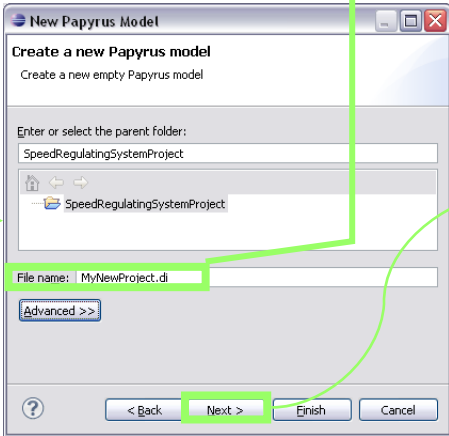
- Within the project explorer view:
  - Select a project > Right click on it > New > Other

**1**



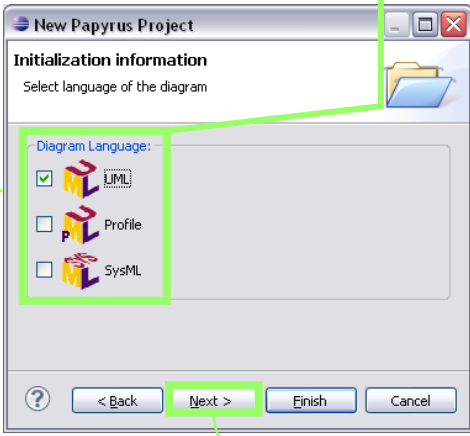
**2**

Enter a model name and press Next.



**3**

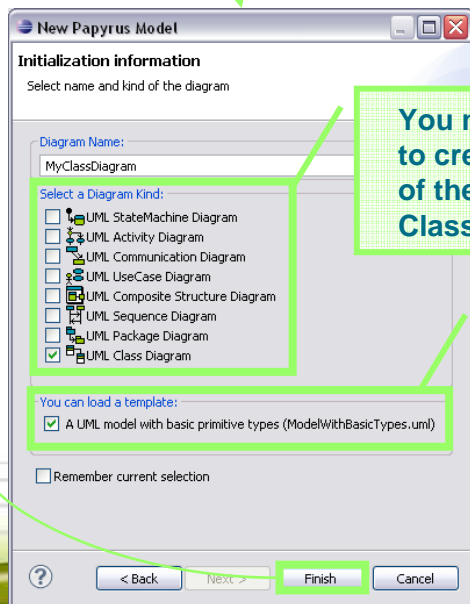
Select a modeling language (e.g. UML)



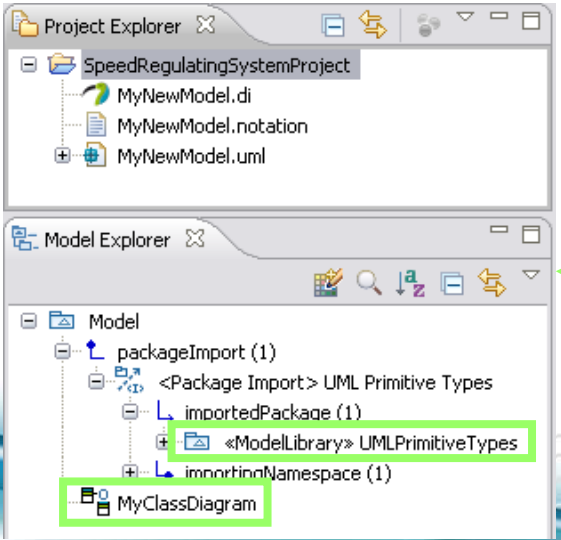
**4**

You may select one diagram to create at the initialization of the project (e.g. a UML Class Diagram).

You may also select model template (e.g. Model Template importing UML Basic data type).



**5**

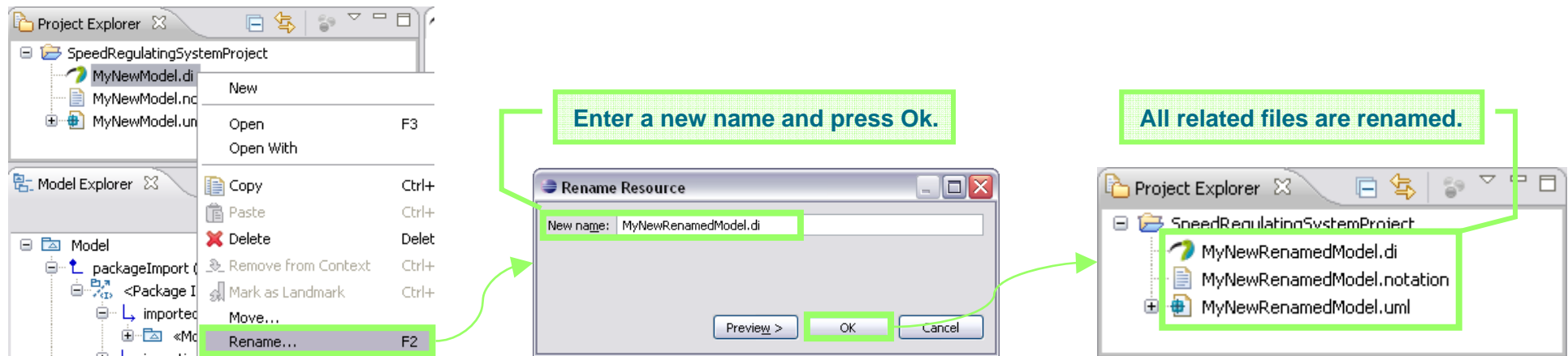


cealist



# Model File Renaming

- **Within the project explorer:**
  - Select the model to rename
  - Right-click on it > Rename (short cut → F2)

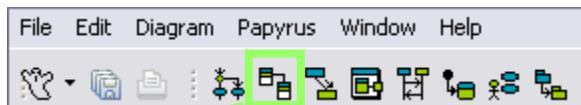
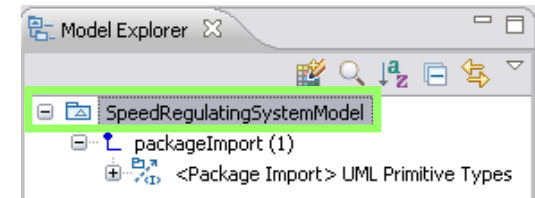




# Diagrams creation

- E.g., creating new class diagram

- Within the model explorer, select the model element that will host the new diagram
- For creating a class diagram:
  - Scenario 1: in the Papyrus tool bar, click on the diagram to create.



New UML Class diagram

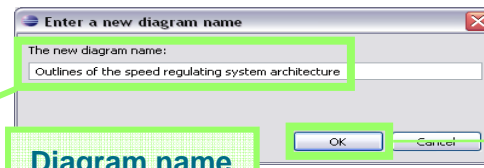
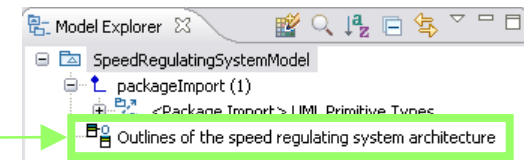
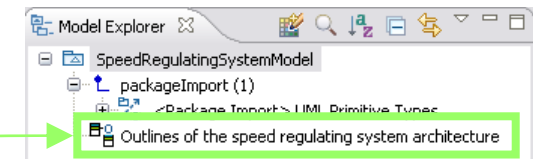
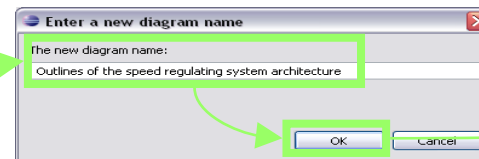
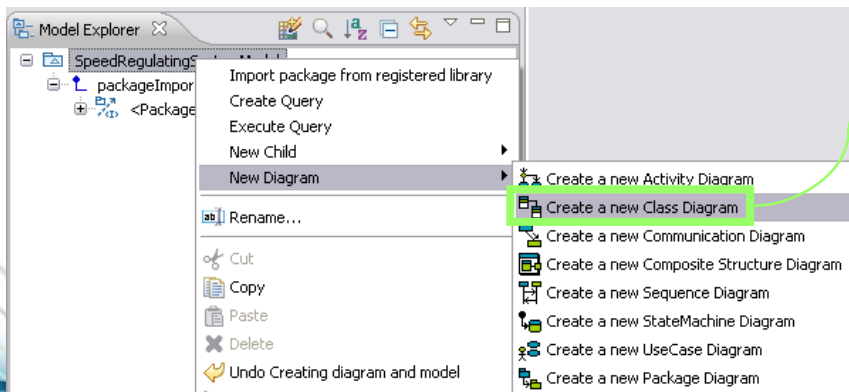


Diagram name



- Scenario 2: left-click on the selected element > New Diagram > Create a new Class Diagram

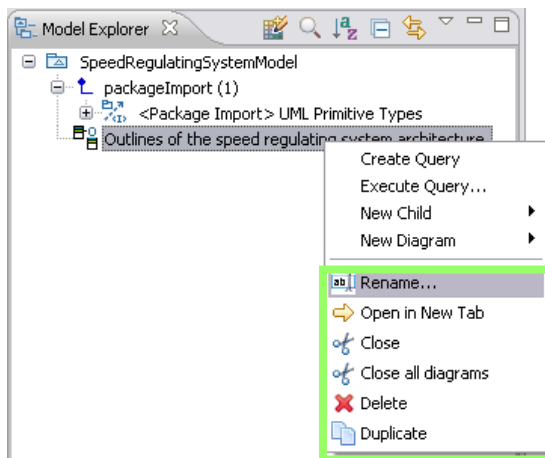




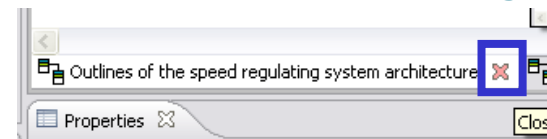
# Diagrams lifecycle management

- **Diagrams can be:**
  - Renamed
  - Closed
  - Open in a new tab
  - Deleted
  - Duplicated
  - Moved from holder to a new one in the model explorer

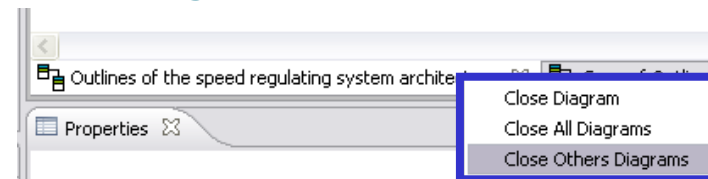
- **Scenario 1: right-click on it in the model explorer > select a command.**



- **Scenario 2: click on the cross located on left-side of the tab of a diag. to close it.**



- **Scenario 3: right-click on the diag. tab for accessing additional close actions:**



- **Moving one diagram in model explorer**

- Within the model explorer, drag and drop the diagram from its origin place into its targeted place.



# Some general details on the model editor

Note: "\*" means the model has been modified since last save.

Palette: enables model elements creation for a given diagram kind (e.g., UML2 class diagram).

Editor tab: within one editor tabs, several diagrams may be open.

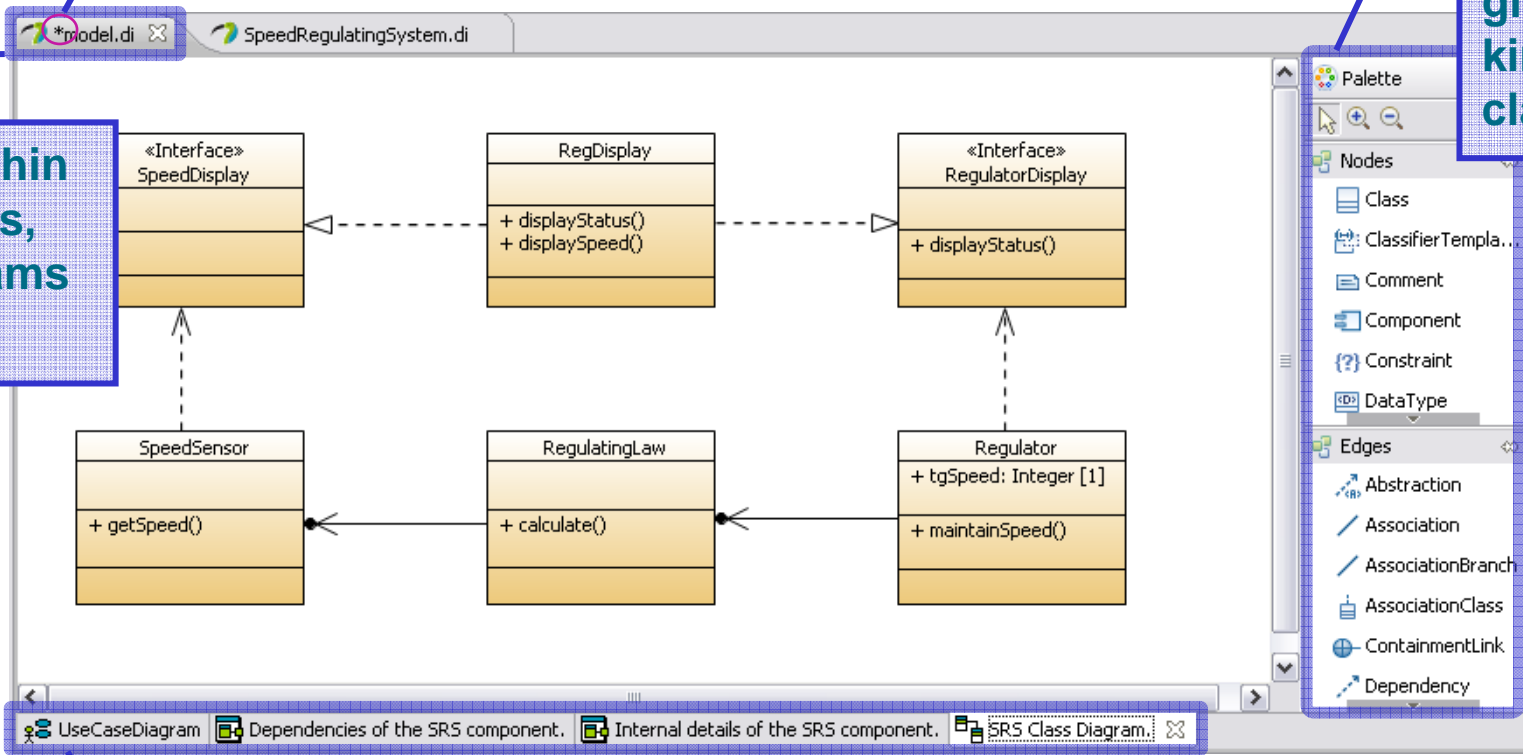


Diagram tabs: each tab holds for one open diagram (e.g., 4 diagrams are open).

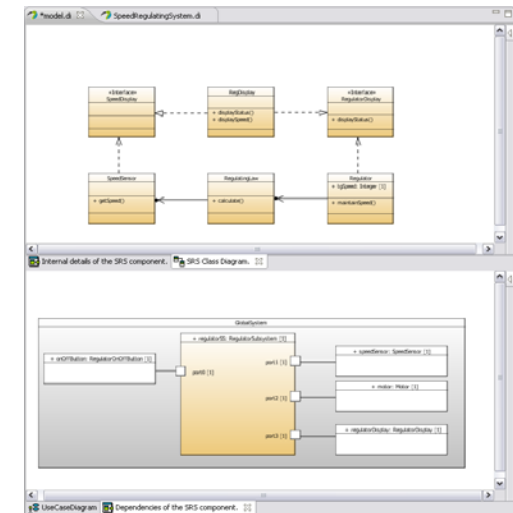
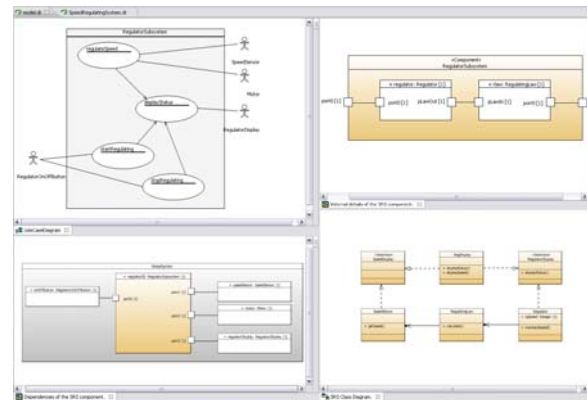
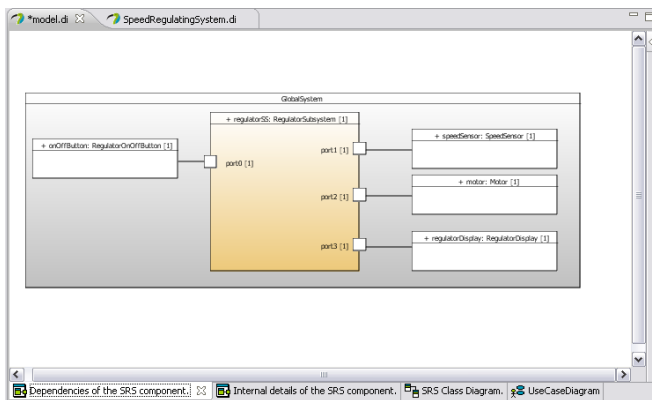




# Generics on Papyrus editors

## • Sash editor facilities

- Enable organizing various diagram editors within one model editor tab.
  - Scenario:
    - Select the diagram,
    - Click on its tab,
    - Drag&drop it on the place you want to show it.



## • Graphical editors are made of two element kinds

- Nodes
  - E.g., Class, Lifeline, State.
- Edges
  - Associations, Message, Transitions.

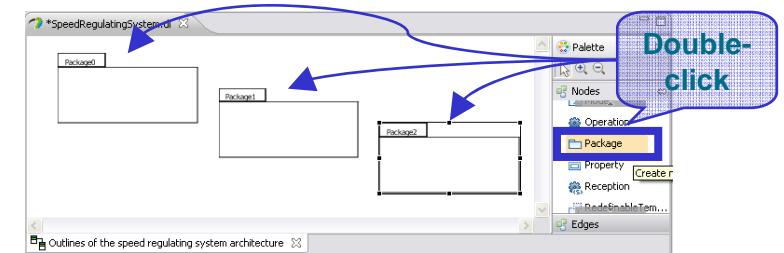
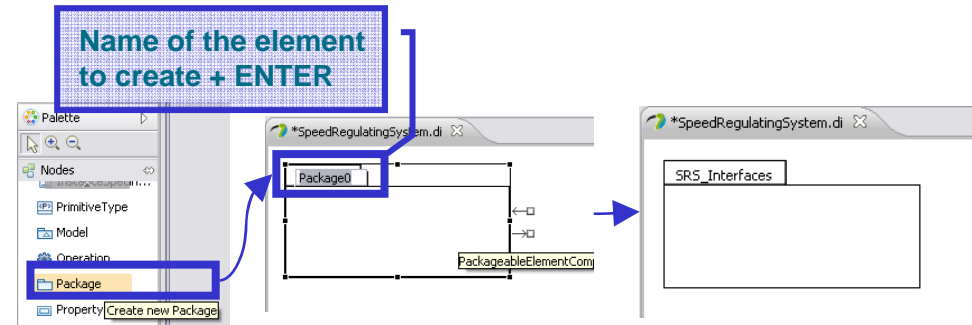
Now, let's see how to populate a diagram...



# Creating nodes

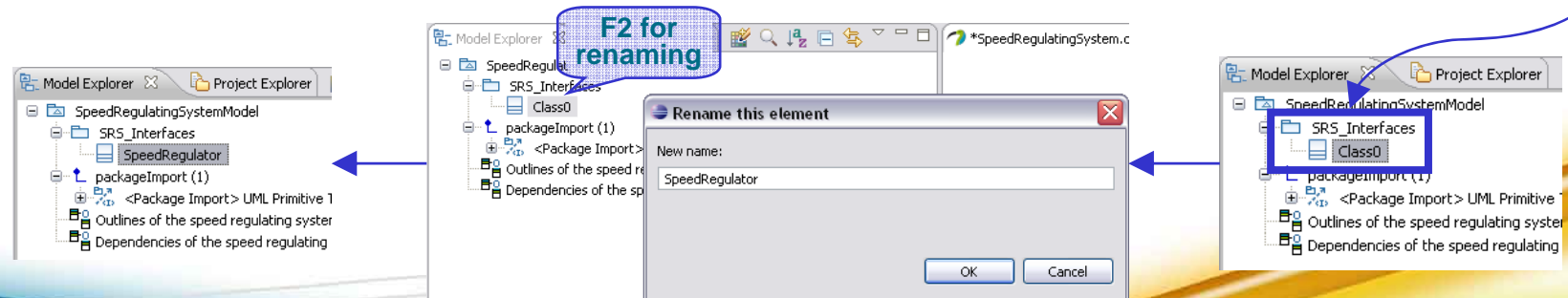
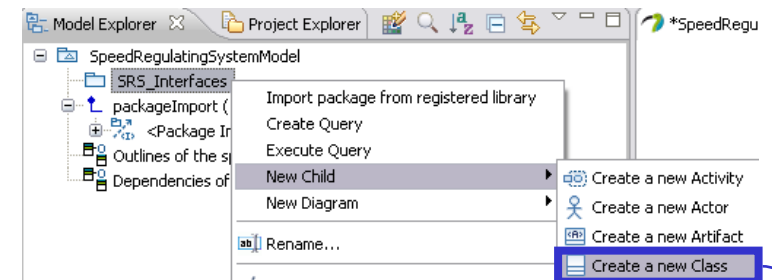
## Using the palette

- Scenario 1:
  - Within the palette, click the kind of element to create.
  - Click within the diagram editor frame where you want to create the model element.
  - Enter a name and press Return.
- Scenario 2 (for creating several model elements):
  - Within the palette, double-click the kind of element to create as many times as you want to create model elements.



## Using the model explorer

- Scenario 1:
  - Within the model explorer, right-click on the model element that will contain the element to create.
  - Select New Child and then select the kind of model element to create.
  - To rename the created element, select it and either press F2 or right-click and select Rename.




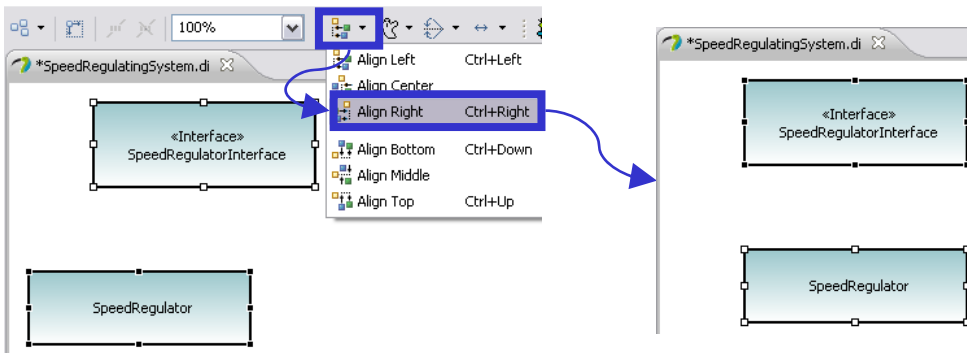



# Graphical alignments of model elements

## • Aligning node elements

### ▪ Scenario 1:

- Select the nodes to align,
- In the tool bar, select the button , and then select one available alignment policy.



 It is the last selected element that is used as reference position!

### ▪ Scenario 2:

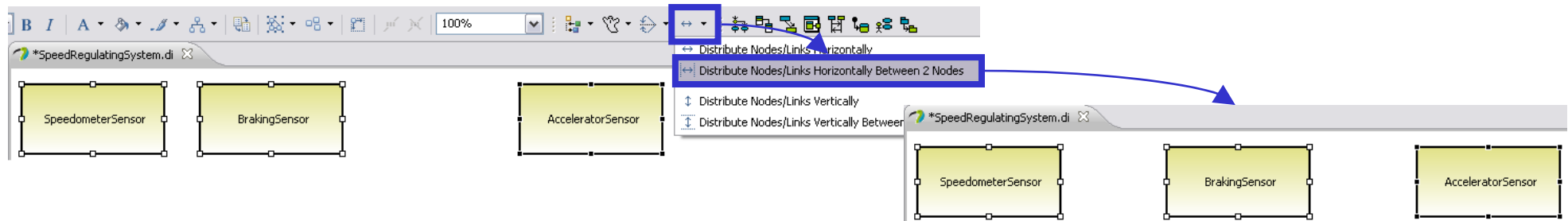
- Select the nodes to align,
- Then, hit keys **Ctrl** + Arrow (←, →, ↑ or ↓).



# Graphical distributions of nodes

- **Scenario:**

- Select the nodes to distribute,
- Apply one of the distribution strategies available from the Papyrus action bar.



- **Notes**

- Two kinds of distribution are possible for both horizontal and vertical directions
  - or : nodes are distributed between both most external selected nodes.
  - or : nodes are distributed in the range of their container
- Example on ports within the composite class diagram

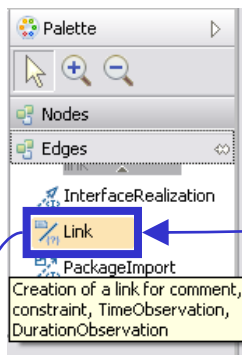


# Adding a Comment

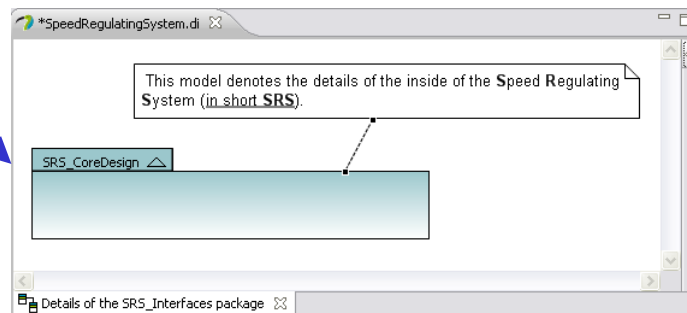
## • Adding a new Comment

### ▪ Scenario:

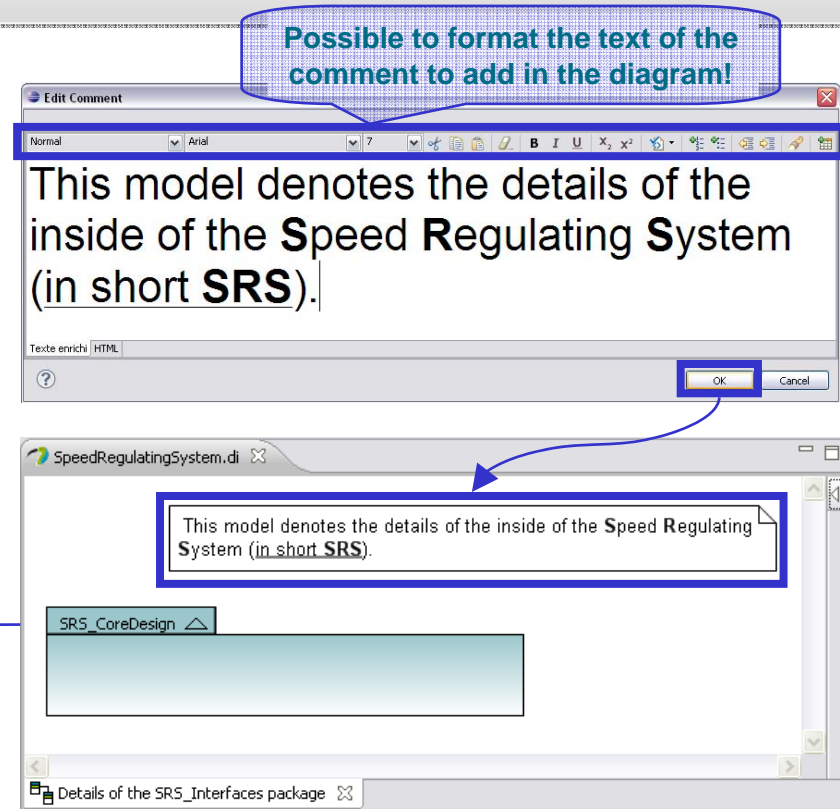
- Add an Comment node on the diagram,
- Type your comment using the enriched textual editor.
- Then, add the links between the new Comment and the elements being element.
  - For that purpose, let's use the tool "Link" in the Palette,



- And draw a link between the created Comment and each element being commented.



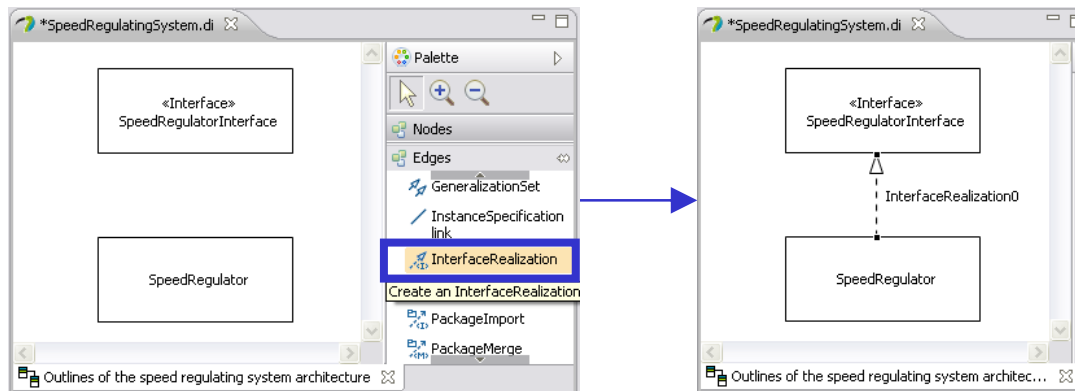
- **Note:** Any kind of model elements, either nodes or edges, may be commented!





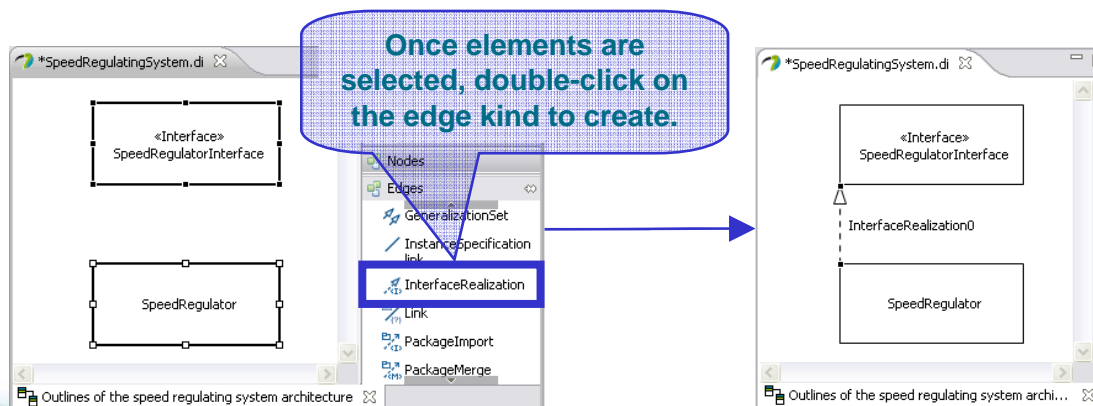
## • Scenario 1:

- Within the palette, click the kind of link to create.
- Within the diagram editor frame, drag and drop the link from its source to its target.



## • Scenario 2:

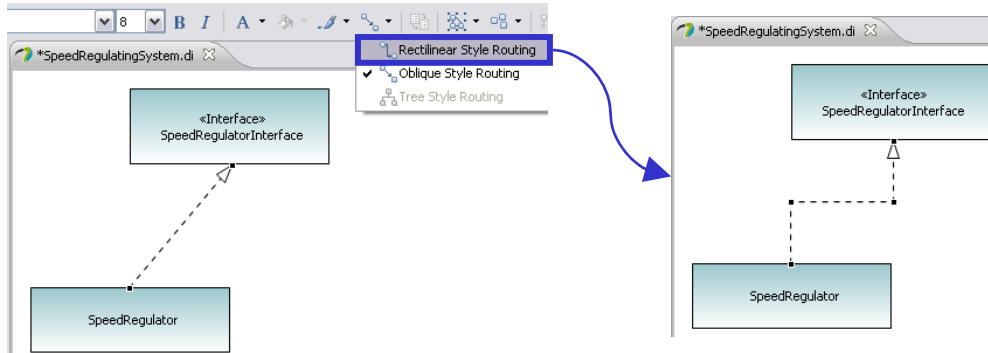
- Within the diagram editor frame, select both source and target elements.
- Next, within the palette, double-click on the edge kind you want to create.





# Routing edge policies

- Oblique versus rectilinear routing policies for edges

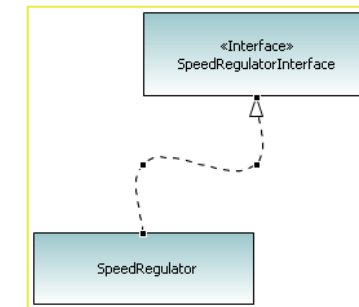
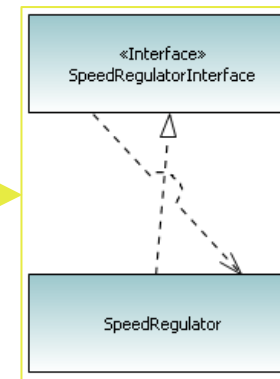
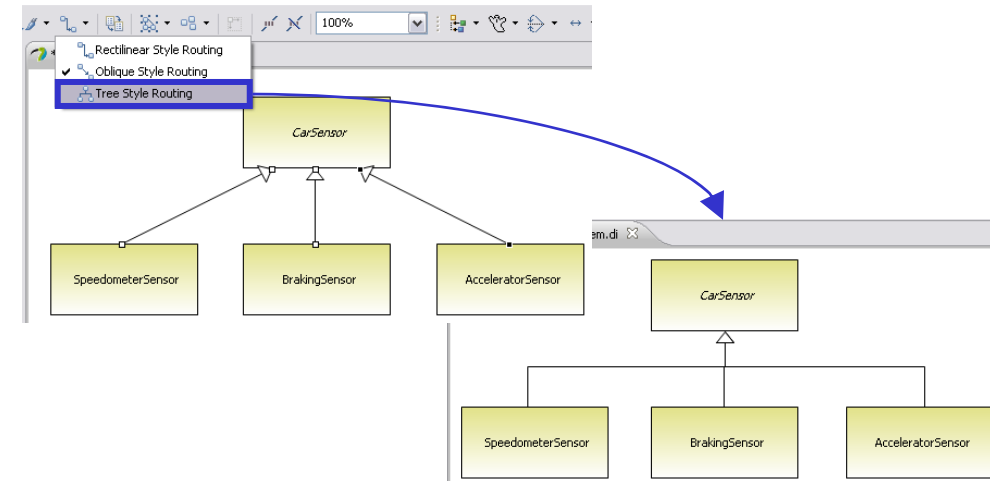


- Possible parameterizations of routing policies within the Appearance tab of the property view:

- Using tree style routing

- Scenario:

- Select the edges to route and apply tree-style routing policy.





# Re-routing Edges

- Using short keys

  - Scenario 1:

    - Select the edges to reroute,
    - Hit (Ctrl + < ↑, ↓, ← or → >) → only opposite nodes move.

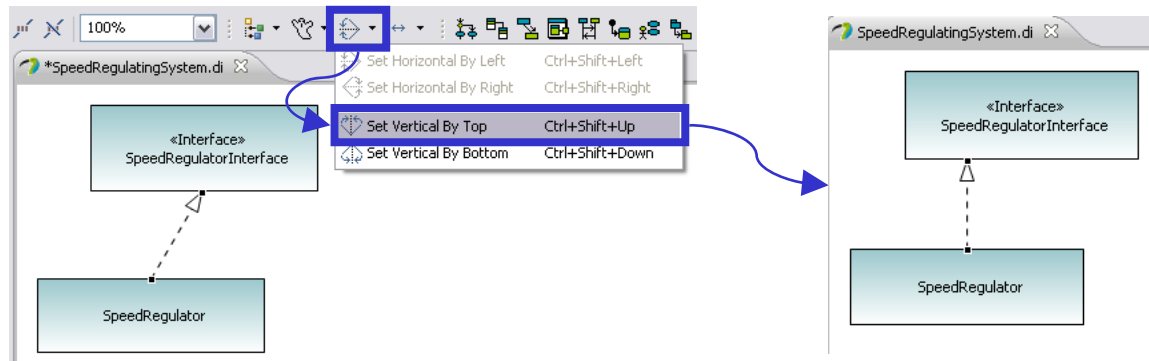
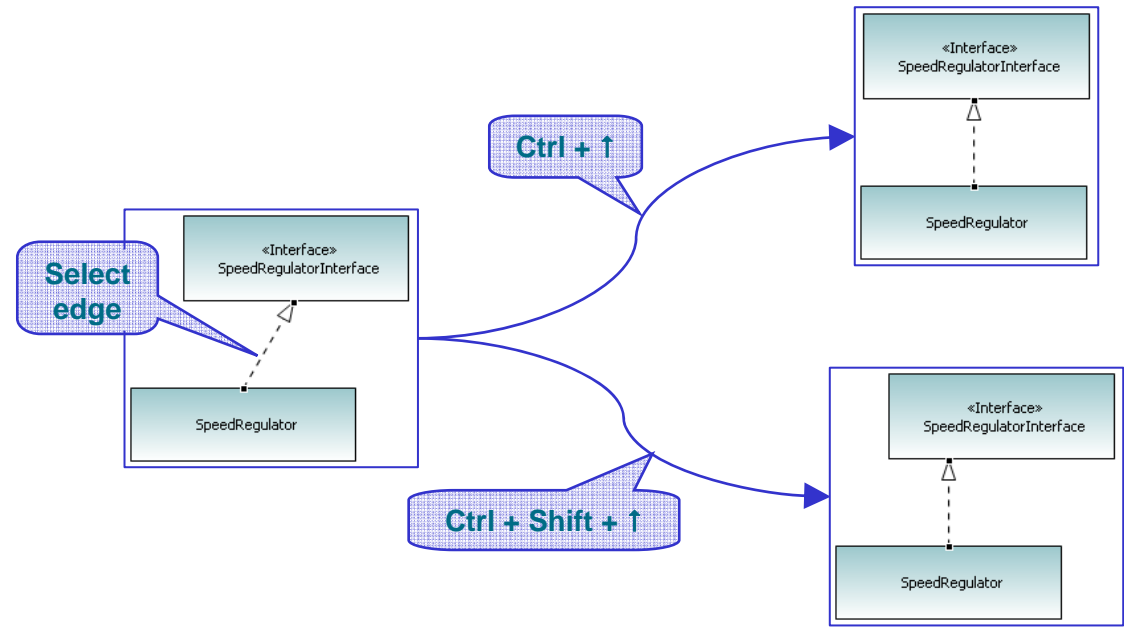
  - Scenario 2:

    - Select the edges to reroute,
    - Hit (Ctrl + Shift + < ↑, ↓, ← or → >) → only edge anchors move.

- Using Papyrus tool bar

  - Scenario:

    - Select the edges to reroute,
    - Select on the command of the menu



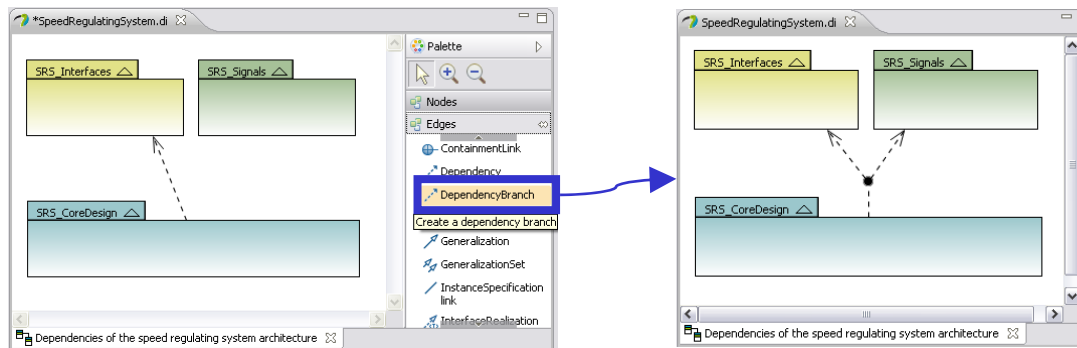




# Modeling multi-branches edges

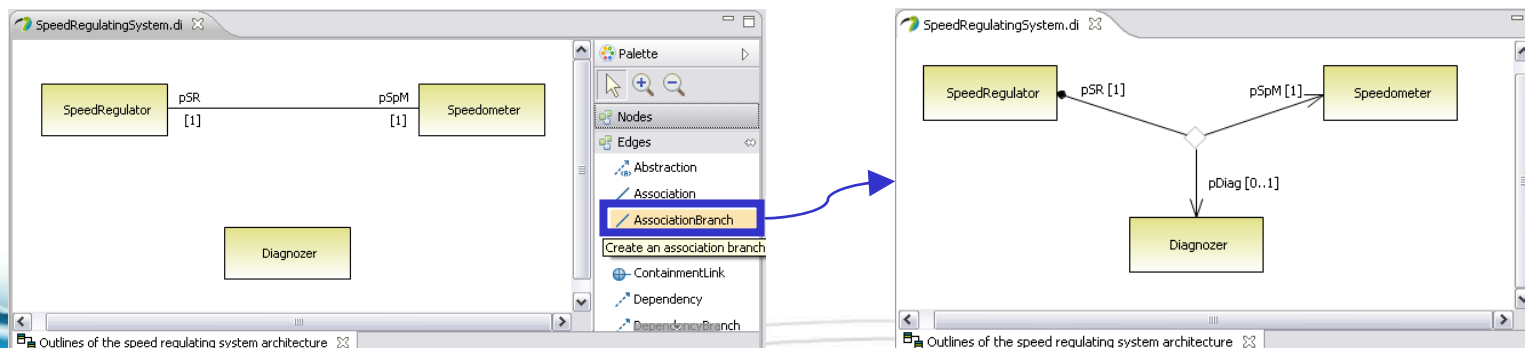
## • Modeling multi-dependencies

- Create a dependency between two of the elements to link
- Add a branch using the tool "DependencyBranch" in the palette.
  - Either from the dependency to the element, if this latter has to be added in the list of source element of the dependency,
  - Or from the element to the dependency, if it has to be added as a target.



## • Modeling multi-associations

- Create an association between two of the elements to link
- Add a branch using the tool "AssociationBranch" in the palette.

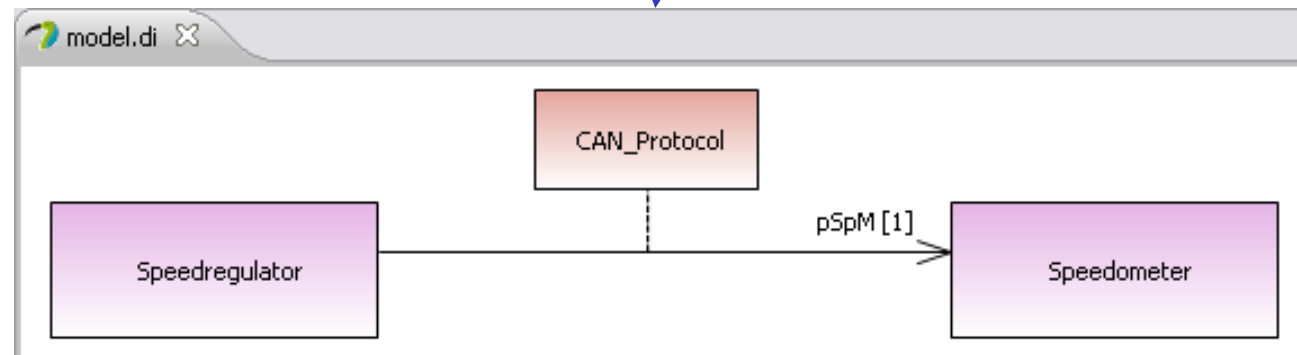
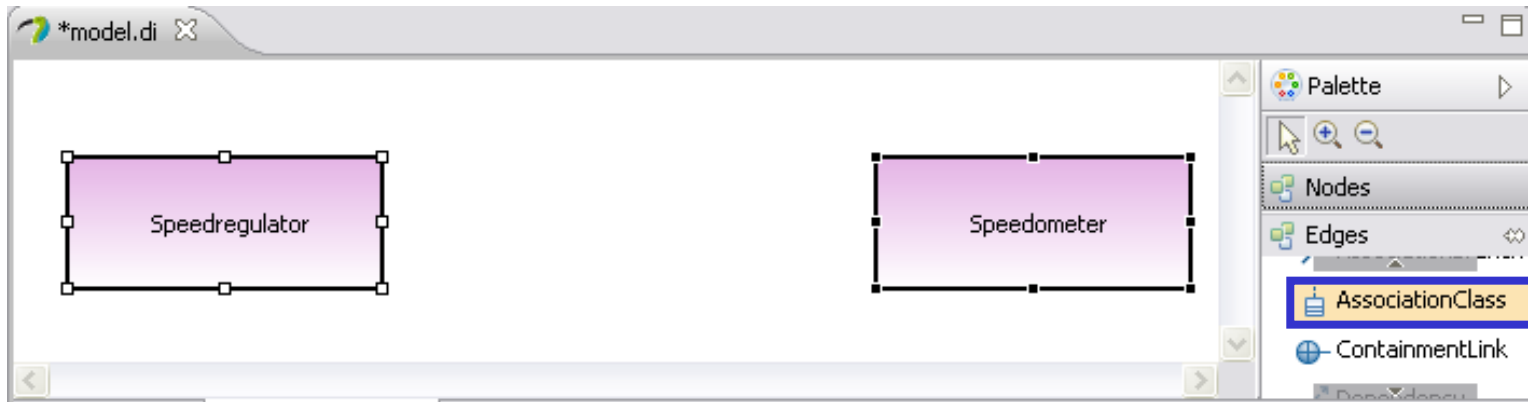




# Modeling ClassAssociations

- **Scenario:**

- Within the palette, use the tool "ClassAssociation".

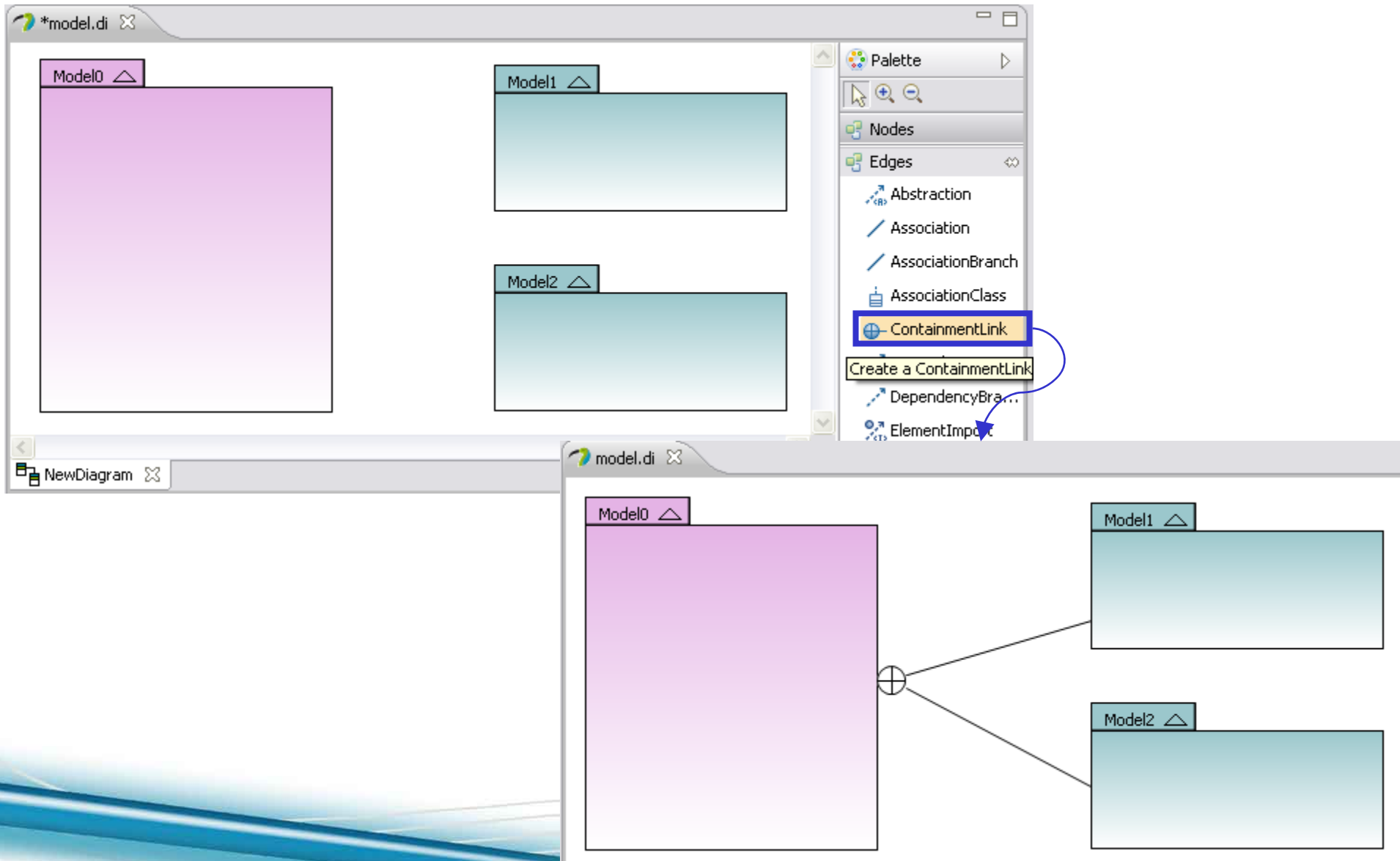




# Modeling containment relationship

- Scenario:

- Within the palette, use the tool "ContainmentLink"



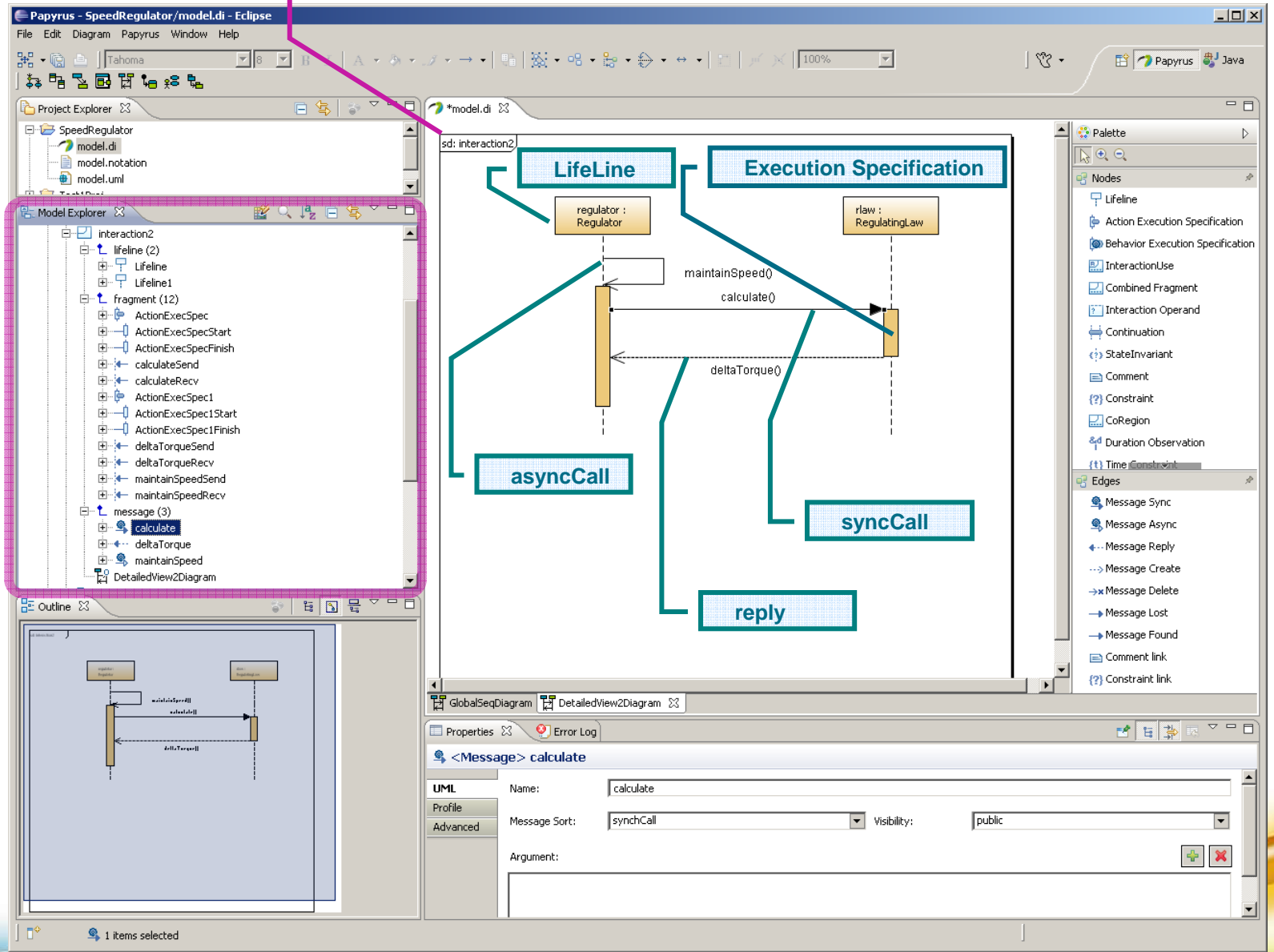


- **Short Overview of Interaction elements**
- **Creating a first basic sequence diagram**
  - Lifelines
  - Execution Specification
  - Messages
- **Structuring Scenarios**
  - Combined Fragments
  - Creation process on a Loop CF
  - From single operand to several the Alt example
- **Setting temporal information on diagrams**
  - Introduction
  - Setting Duration Constraint example



# My first papyrus UML Sequence diagram - basics

Sequence Diagram of interaction



## Model elements of interactions

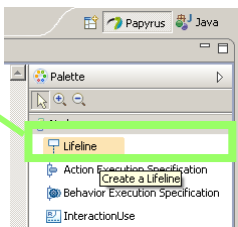
- Lifelines
- Fragments
  - Execution Specifications
  - Events
  - Combined Fragments
- Messages
  - asyncCall
  - sync
  - reply
  - create
  - delete



# UML Sequence diagrams : basics - Lifeline creation

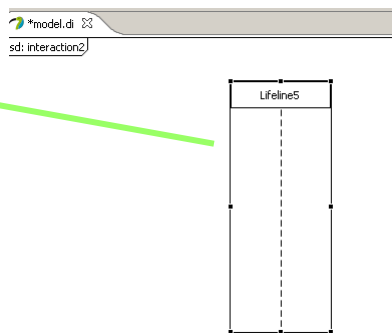
1

Select Lifeline tool in the palette



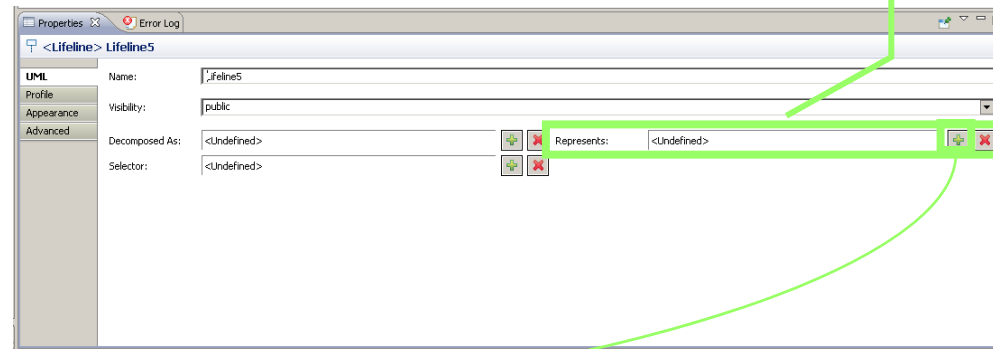
2

Click in the diagram to drop the lifeline



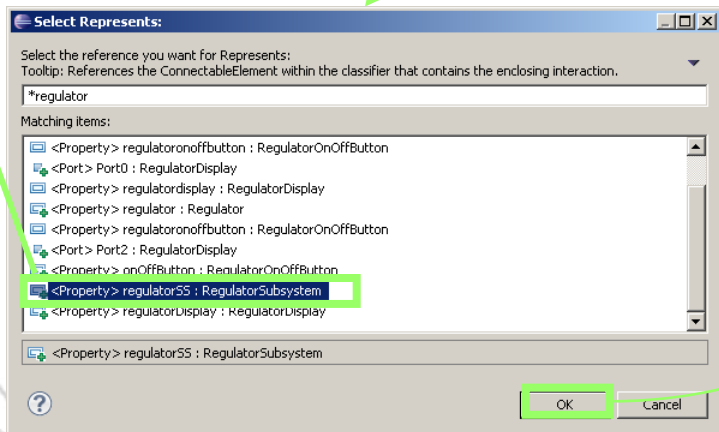
3

Set the represents property : click on the red cross then select a part in the pop-up



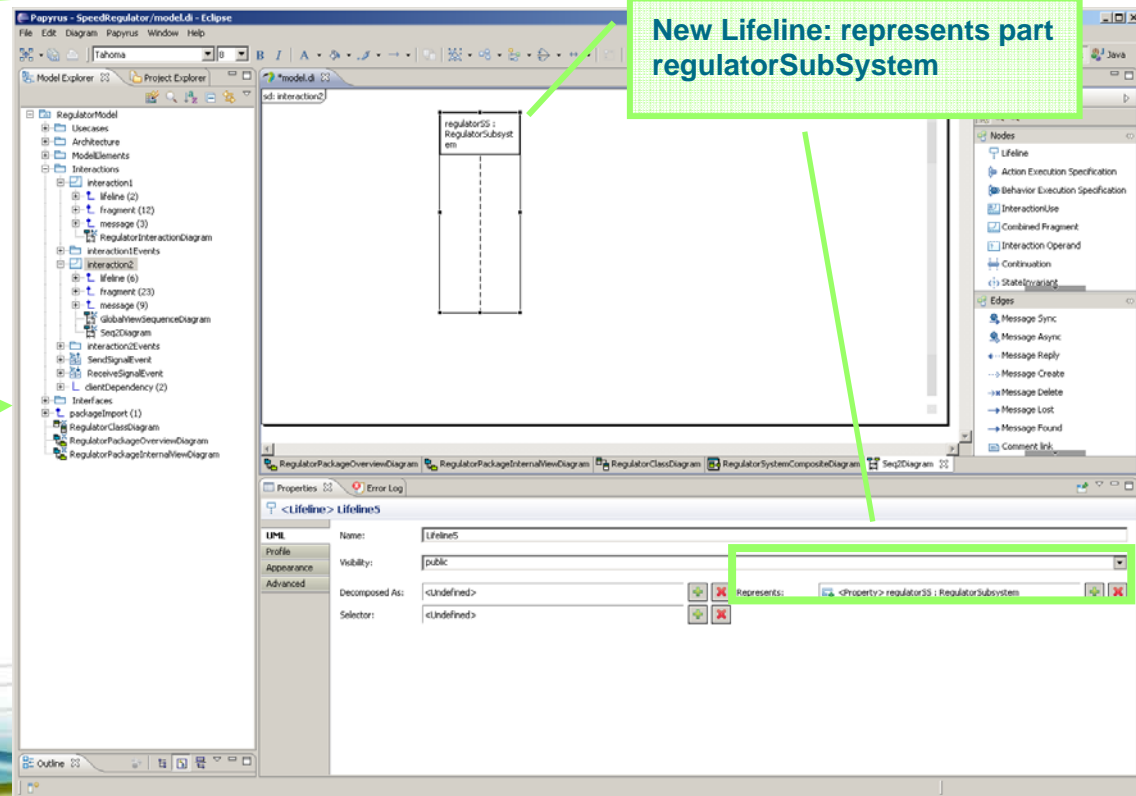
3

Select a part in the pop-up and press OK



4

New Lifeline: represents part regulatorSubSystem





## Two kinds of Execution Specification can be created

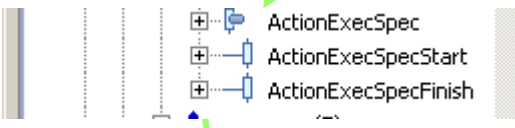
- Action ES
- Behavior ES

Select ES tool in the palette

1

Click and drop the ES on the lifeline.

2



**New created UML Elements:**

- ActionExecutionSpec
- 2 events
  - ActionExecutionSpecStart
  - ActionExecutionSpecFinish

regulatorSS : RegulatorSubsystem

Properties: <Action Execution Specification> ActionExecSpec

UML Name: ActionExecSpec  
 Profile: public  
 Visibility: public  
 Association: <Undefined>  
 Start: <Execution Occurrence Specification> ActionExecSpecStart  
 Finish: <Execution Occurrence Specification> ActionExecSpecFinish

- Papyrus MDT provides dynamic support for message creation

- Selection of operation or signal attached to message
- Dynamic creation of operation/signal
- Dynamic creation of ES for synchronous message

1

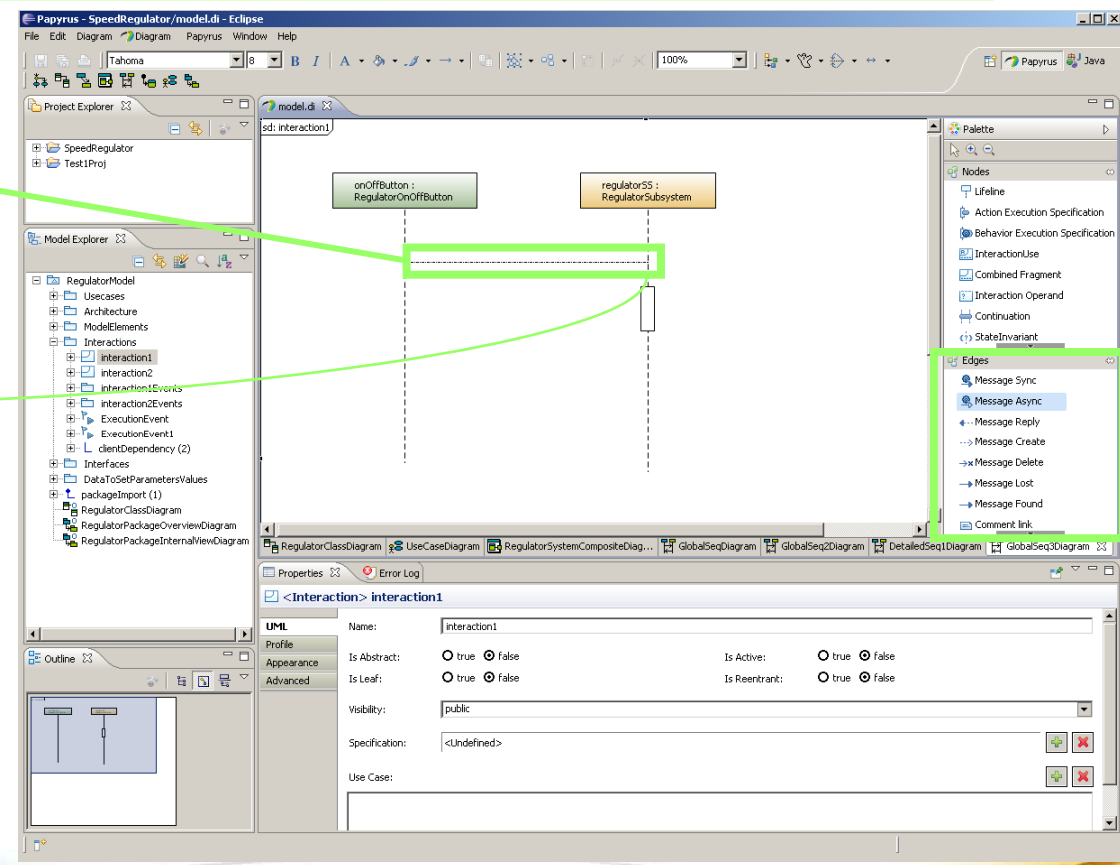
Select Message Creation tool from palette.

2

Place the message on the diagram click to set source, maintain mouse button down an drag towards target then release

3

Select an operation or signal or create a new one







## Hints and restrictions for message creations :

### 1. ASync signal

- Async signal is not provided in the palette, actually they are async messages with messageSort property set to asyncSignal (in the property view)
- The property is set automatically when a signal is selected from the pop-up menu

### 2. Sync message

- A sync message can be defined only if it starts from an ES.
- A target ES is created automatically if the target anchor point is not an ES.

### 3. Create message

- A create message can be defined only between two existing lifelines

### 4. Delete message

- A delete message can be defined only towards the position of a destructionEvent

### 5. Reply message

- A reply message can be created only from an ES created by a Sync Message



# UML Sequence diagrams : Combined Fragments - overview

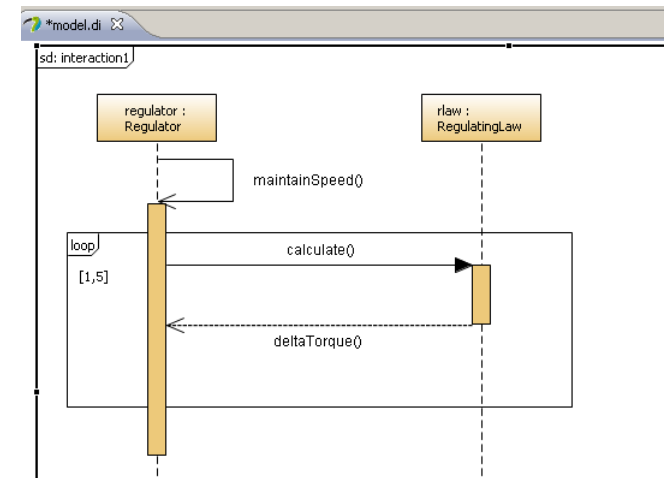
- **Papyrus MDT provides support for combined fragments**
  - Combined fragments represent sub-scenarios
  - They are represented as a rectangle area covering part of a sequence diagram
    - This area can be splitted in several sub-areas corresponding to operands
      - for instance in the case of the alt CF (alternatives) which represents a choice of behaviors)
  - They can be assembled to represent generic complex behaviors of a system
- **Combined Fragments supported are :**
  - alt, opt, par, loop, break, critical, neg, assert, seq, strict, ignore, consider

- **A Combined Fragment...**

- Covers Lifelines,
  - It represents a sub-scenario involving the covered lifelines
- Has one or more operands,
  - Loop, break, neg, assert, opt have exactly one operand
- And has gates to connect incoming/outcoming messages

- **Creating a Combined Fragment consists in 6 steps**

1. select the CF tool in the palette
  2. select the type of combined fragment consider/ignore or other
  3. place the CF on the diagram
  4. select the type of Interaction operator (by default a Seq CF is created).
  5. create the operands if necessary (by default one is created)
  6. set operand properties in the property view
- Steps 4., 5., 6. vary according to the interaction operator selected and specific rules may apply.





# UML Sequence diagrams : Combined Fragments Creation(1)

- Creation steps 1, 2, 3 , 4 (the loop example)

**1**

Select CF tool in the palette

**2,3**

Drop the CF and draw the rectangle on the lifelines. Then select kind of CF Consider/Ignore or Standard CF

New CF with default operand kind = Seq

**4**

Change operand to Loop:  
- Click within the operand area  
- Select Loop kind in the properties view

sd: interaction1

```

sequenceDiagram
    participant regulator as regulator : Regulator
    participant rlaw as rlaw : RegulatingLaw
    regulator->>regulator: maintainSpeed()
    activate regulator
    regulator->>rlaw: calculate()
    activate rlaw
    rlaw-->>regulator: deltaTorque()
    deactivate rlaw
    deactivate regulator
  
```

regulator : Regulator

rlaw : RegulatingLaw

maintainSpeed()

calculate()

deltaTorque()

loop

MyLoopCF

Interact...perator: loop

Visibility: public

loop  
critical  
neg  
assert



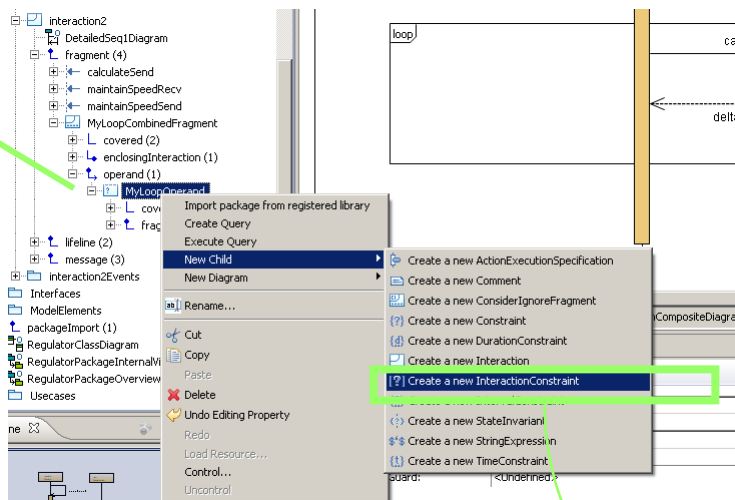
# UML Sequence diagrams : Combined Fragments Creation (2)

## • Creation steps 5, 6 Setting Guard of the operand

- Create Interaction constraint
- Set Guard constraint in the properties view

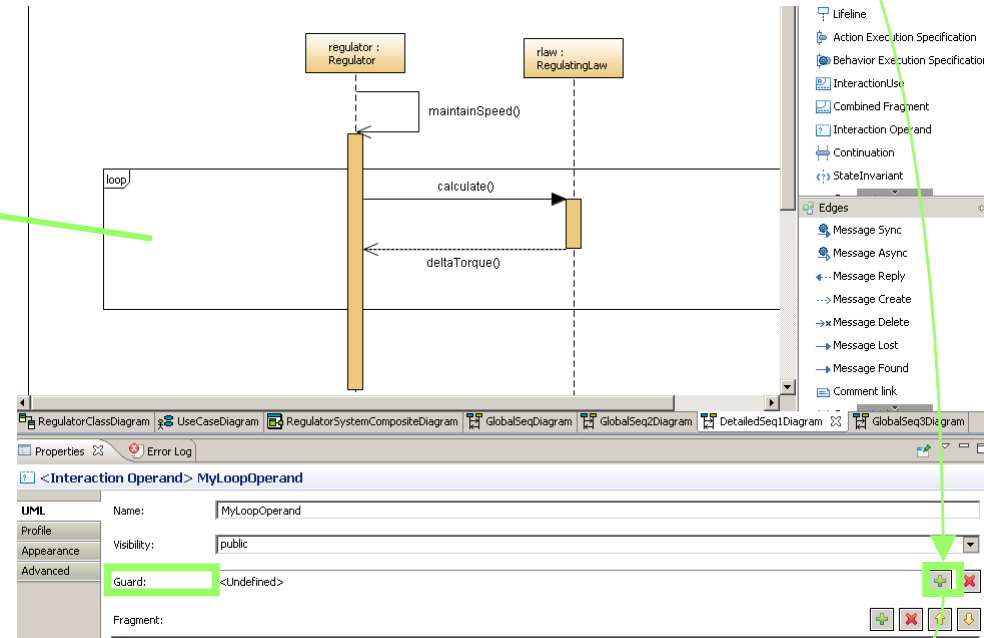
5

**In the model explorer  
Create an interaction  
constraint**

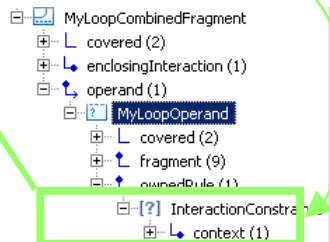


6

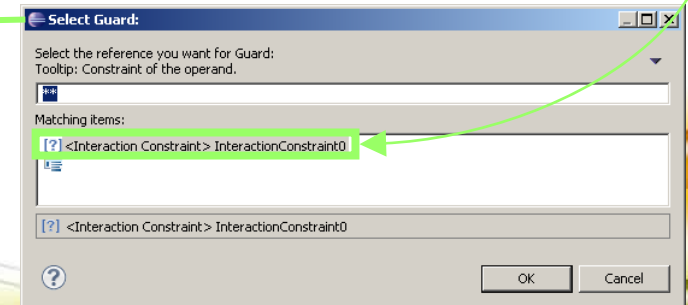
**Set Guard property of Loop Operand**  
- Click in the operand area then ...  
- In the properties view select guard (click on green "+" sign)



**New CF with  
default operand  
kind = Seq**



**Set Guard property of Loop  
Operand**  
- Select Interaction Constraint  
- Press OK





# UML Sequence diagrams : Combined Fragments Creation (3)

## • Creation steps 6 Setting Guard of the operand (cont)

- Setting properties of the guard (Min, Max and Specification)

## • Current limitation

- The current implementation of properties view does not allow dynamic creation of elements
- So we have to temporarily use a turn around

Procedure to follow

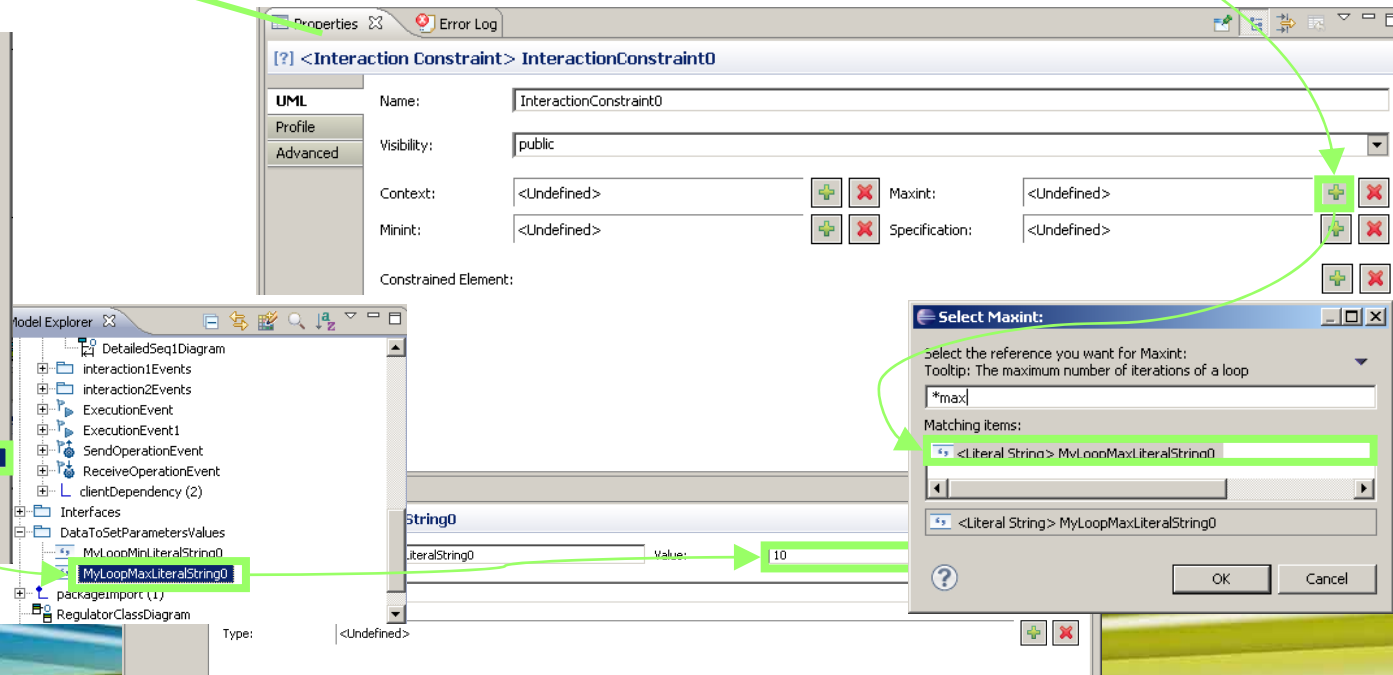
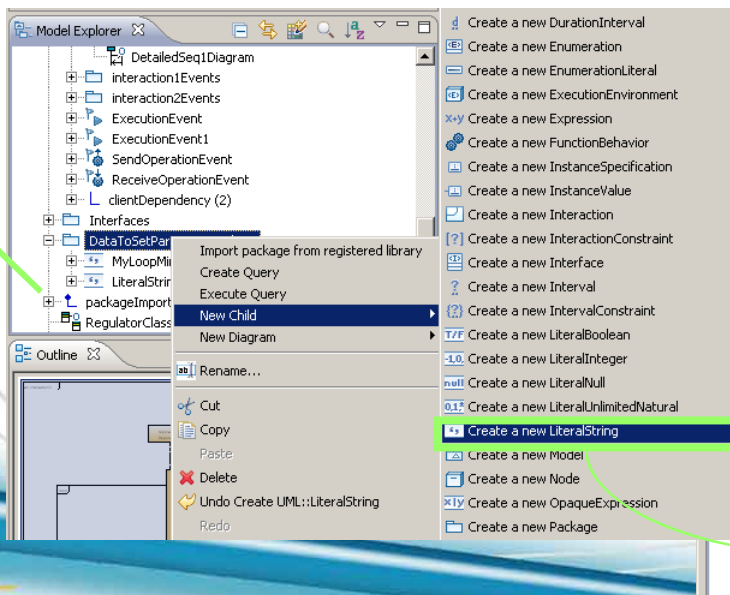
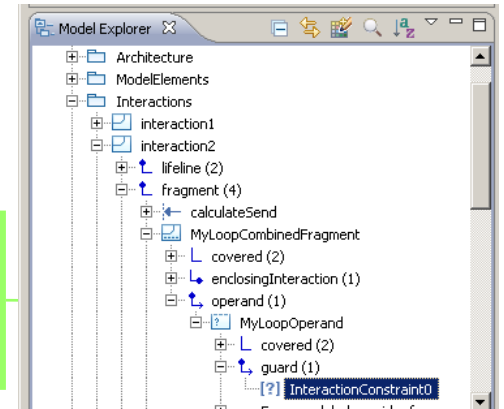
1. create values separately in the model explorer
2. reference these values in the properties view

7

In the model explorer  
Create a package to create values  
that will be referenced by the guard:  
- Min  
- Max

8

Set Guard properties of Loop Operand  
- in model explorer select guard constraint  
- in properties view select property to set  
(click on green "+" sign on the right)





# UML Sequence diagrams : Combined Fragments Creation (4)

When all properties are set, the guard is displayed in the Combined Fragment

The screenshot displays the Eclipse IDE interface for creating a UML sequence diagram. The main diagram area shows a sequence diagram with two lifelines: 'regulator : Regulator' and 'rlaw : RegulatingLaw'. A loop is defined with a guard '[2,10]'. Inside the loop, the 'regulator' lifeline sends a 'maintainSpeed()' message to itself, then a 'calculate()' message to 'rlaw', which returns 'deltaTorque()' to 'regulator'. The Properties view at the bottom shows the configuration for 'InteractionConstraint0', with 'Minint' set to '<Literal Integer> min' and 'Maxint' set to '<Literal Integer> max'. The Model Explorer on the left shows the diagram's structure, with 'InteractionConstraint0' selected under the 'guard' element.

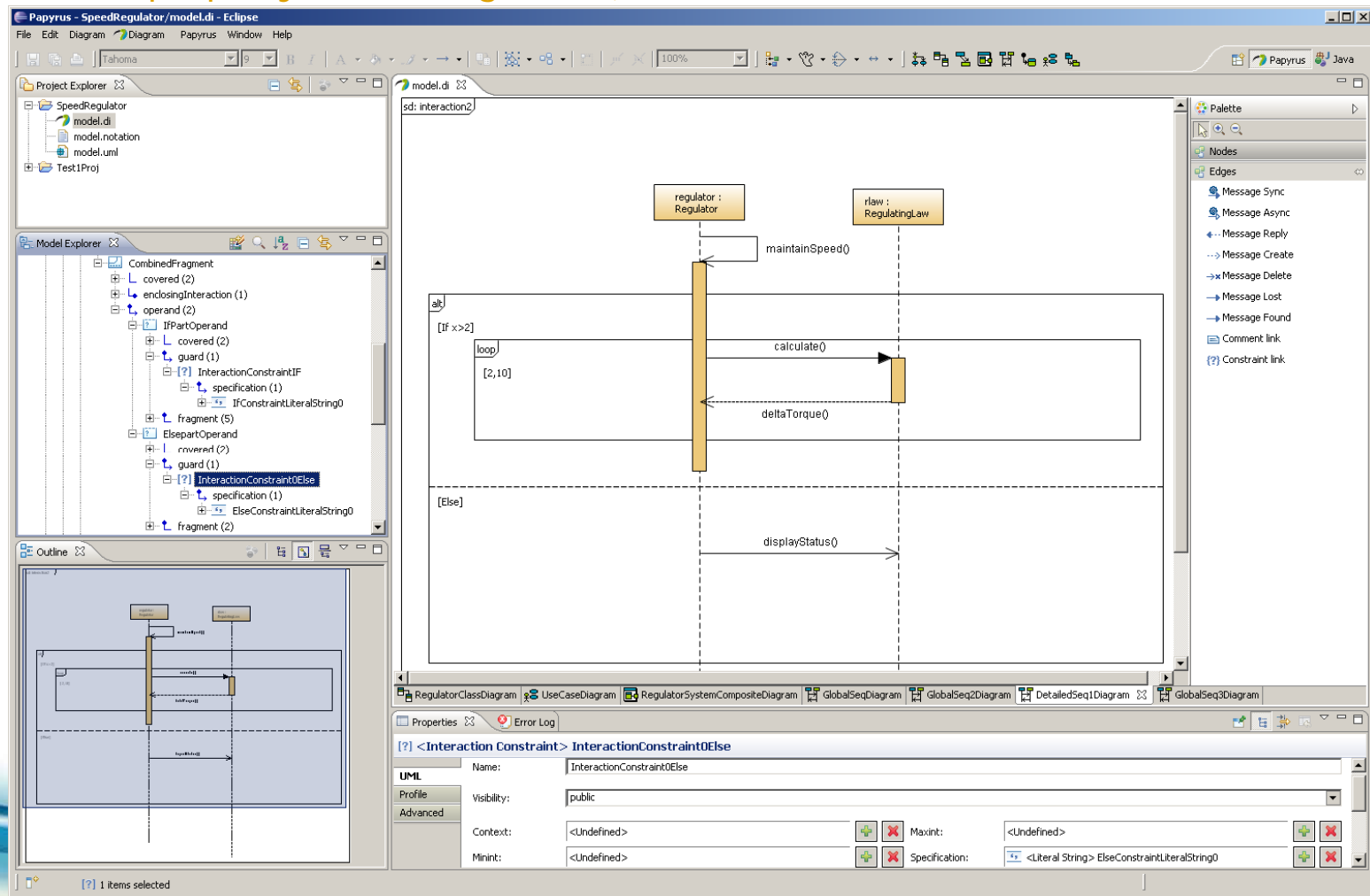
## Remark :

- min and max properties are set via the properties view
- the specification property can be defined directly from the model explorer (using contextual menu on interactionConstraint)



# UML Sequence diagrams : Combined Fragments Creation (5)

- Creating Combined Fragments with more than one operand (the alt example)
- Same process
  - Just select a new operand tool in the palette, then click in the operand area
  - A new area appears with a separation line
  - You can enter guards for each operand in the same manner as above (use specification property as a string literal)





# UML Sequence diagrams : Setting temporal information (1)

- Temporal information on sequence diagrams

- Observations

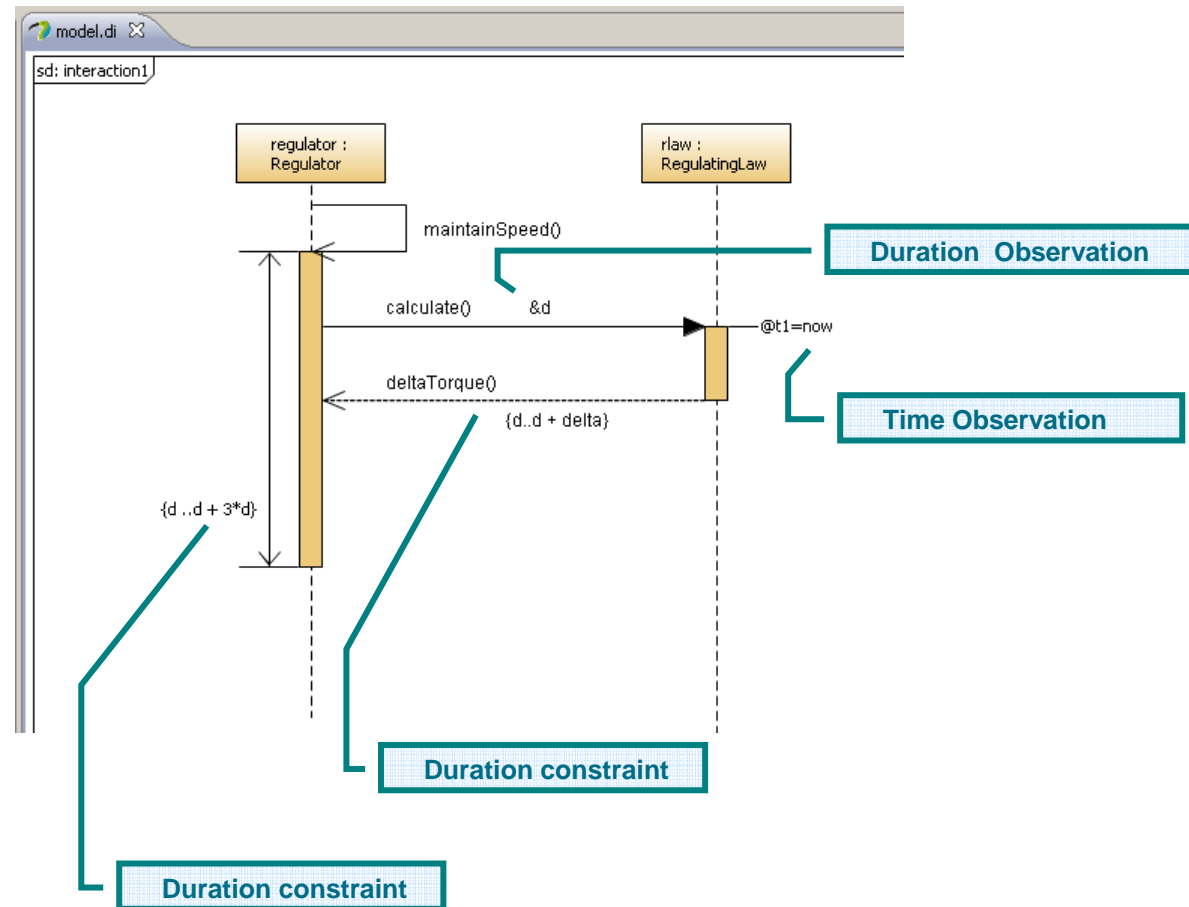
- Time Observation
    - Duration observation

- Constraints

- Time Constraint
    - Duration Constraints

- General procedure to follow

1. Select tool from palette
2. Select anchor point(s) in diagram
3. Release mouse
4. The element is created in the model (with default values set to 0)
5. Set values (depending on the type of element)







# UML Sequence diagrams : Setting temporal information (2)

## • Creating a Duration Constraint

2

Drop the constraint  
-between two events on a lifeline  
- start and end of a message

1

Select Duration Constraint  
tool in the palette

- Nodes
  - StateInvariant
  - Comment
  - Constraint
  - CoRegion
  - Duration Observation
  - Time Constraint
  - Time Observation
  - Duration Constraint**
  - Destruction Event
- Edges
  - Message Sync
  - Message Async
  - Message Reply
  - Message Create
  - Message Delete
  - Message Lost
  - Message Found
  - Comment link
  - Constraint link

3

Set DurationInterval  
values:  
In the model explorer  
  
Select Duration Interval  
- Min  
- Max

RegulatorModel

- UseCases
- Architecture
- ModelElements
- Interactions
  - interaction1
  - interaction2
    - ownedRule (1)
      - DurationConstraint
        - constrainedElement (2)
          - specification (1)
            - DurationInterval
              - min (1)
                - expr (1)
                  - <Literal Integer> 0**
                - max (1)
                  - DurationIntervalMax1
                    - expr (1)
              - context (1)
            - lifeline (2)
            - Fragment (16)
            - message (2)
            - InternalViewSeq2Diagram
          - ExecutionEvent
          - ExecutionEvent1
          - ExecutionEvent2
          - ExecutionEvent3
          - ExecutionEvent4
          - ExecutionEvent5
          - SendOperationEvent
          - ReceiveOperationEvent
          - ExecutionEvent6

4

In properties view  
→ Set value

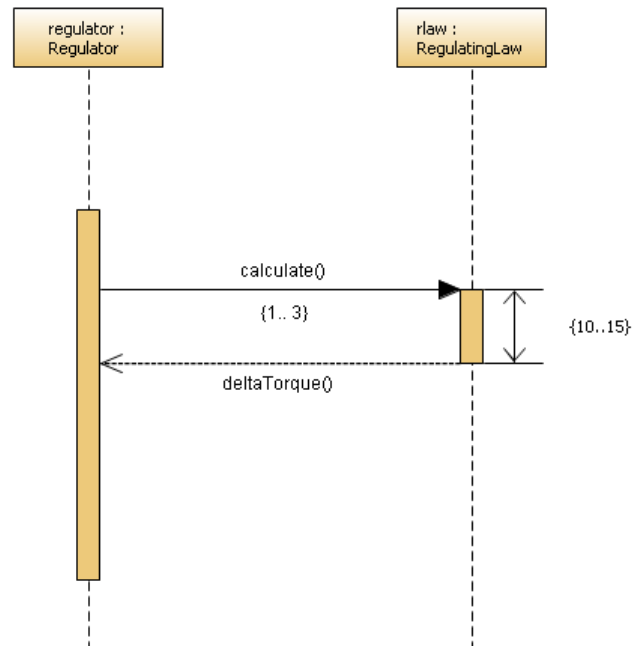
Properties view for **<Literal Integer> 0**

|          |             |             |        |          |
|----------|-------------|-------------|--------|----------|
| UML      | Name:       |             | Value: | <b>0</b> |
| Profile  | Visibility: | public      |        |          |
| Advanced | Type:       | <Undefined> |        |          |



# UML Sequence diagrams : Setting temporal information (3)

Other timing information can be set with the same process



**Remark : Types of attributes vary depending on the element**

- **Duration Constraints** have a **Duration Interval** as specification
  - Two association min and max are of type Duration  
(We use integers to set duration values)
- **Timing Constraints** have a **timeInterval** as specification
  - Two association min and max are of type TimeExpression  
(We use strings to set TimeExpression values)



# UML as a basic support for DSML

- **Originally intended for modeling software-intensive systems**
  - UML models capture different views of a software system (information model, run-time structure/behavior, packaging, deployment, etc.)
  - Inspired primarily by the concepts from object-oriented languages (class, operation, object, etc.)
- **However, the general nature of its concepts made UML suitable for extensions to other domains.**

**Domain Specific Modeling by profiling the UML2!**





- **UML Profile**

- A special kind of package containing stereotypes, modeling rules and model libraries that, in conjunction with the UML metamodel, define a group of domain-specific concepts and relationships.

- **Profiles can be used for two different purposes:**

- To define a domain-specific modeling language.
- To define a domain-specific viewpoint.

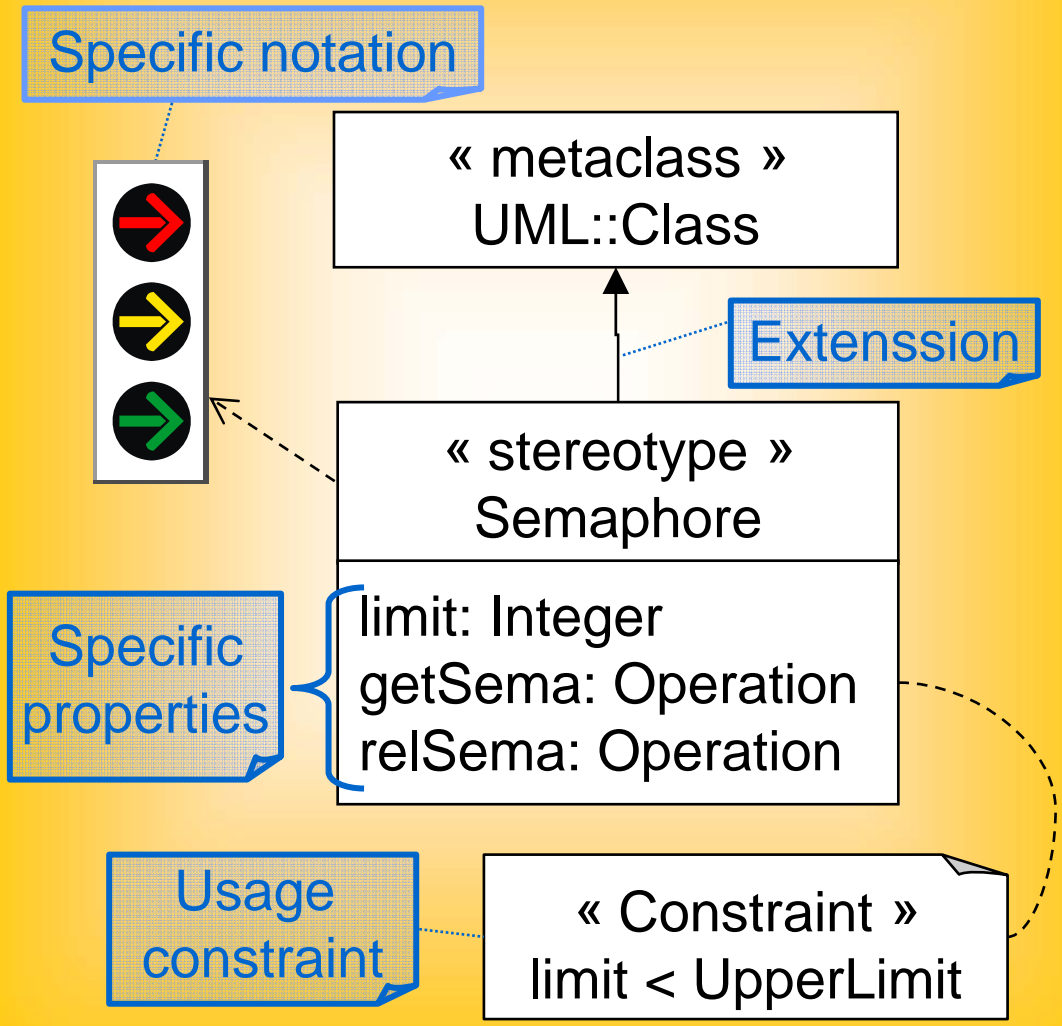
- **Minimal benefits of profile usage are:**

- Correctly defined profiles allow direct and effective reuse of the extensive support structure provided for UML (e.g., Tools, methods, experience, training...).
- DSMLs based on UML profiles share a common semantic foundation which can greatly reduce the language fragmentation problem.

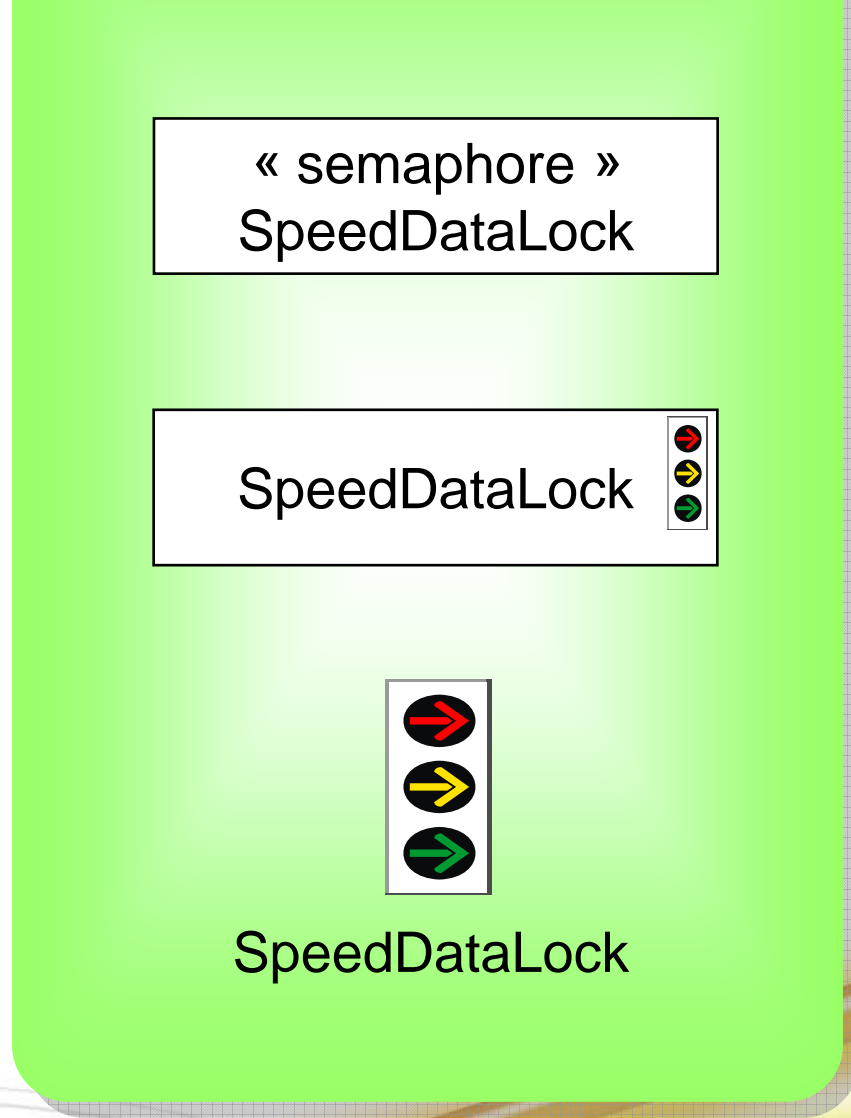


# On UML profiles in one slide!

## Profile definition (Language definition level)



## Profile application (User model level)





- **Main objective:**

- Create your own modeling editor for your domain specific language
- Reduce cost and time to develop such editor
  - Reuse and customize existing editor where possible
  - Benefit from common services (Collaborative work, Resource management...)
  - Shared maintenance on common parts
  - Benefit from existing diagrams
- Ease customization with dynamic configuration tools
  - Allow preview visualization and test

- **A minimal example: SysML-like requirement diagrams**

- Small set of concepts
  - Requirement / Solution / Satisfy link (between Requirement and Solution)
- EMF
  - Not discussed here but Papyrus accepts non UML2 language and diagrams
  - Our customized GMF tooling may also be used
- UML2 Profile
  - Reuse and customize existing diagrams rather than developing new editor
  - Propose user-friendly customization tools



- Language support with profile

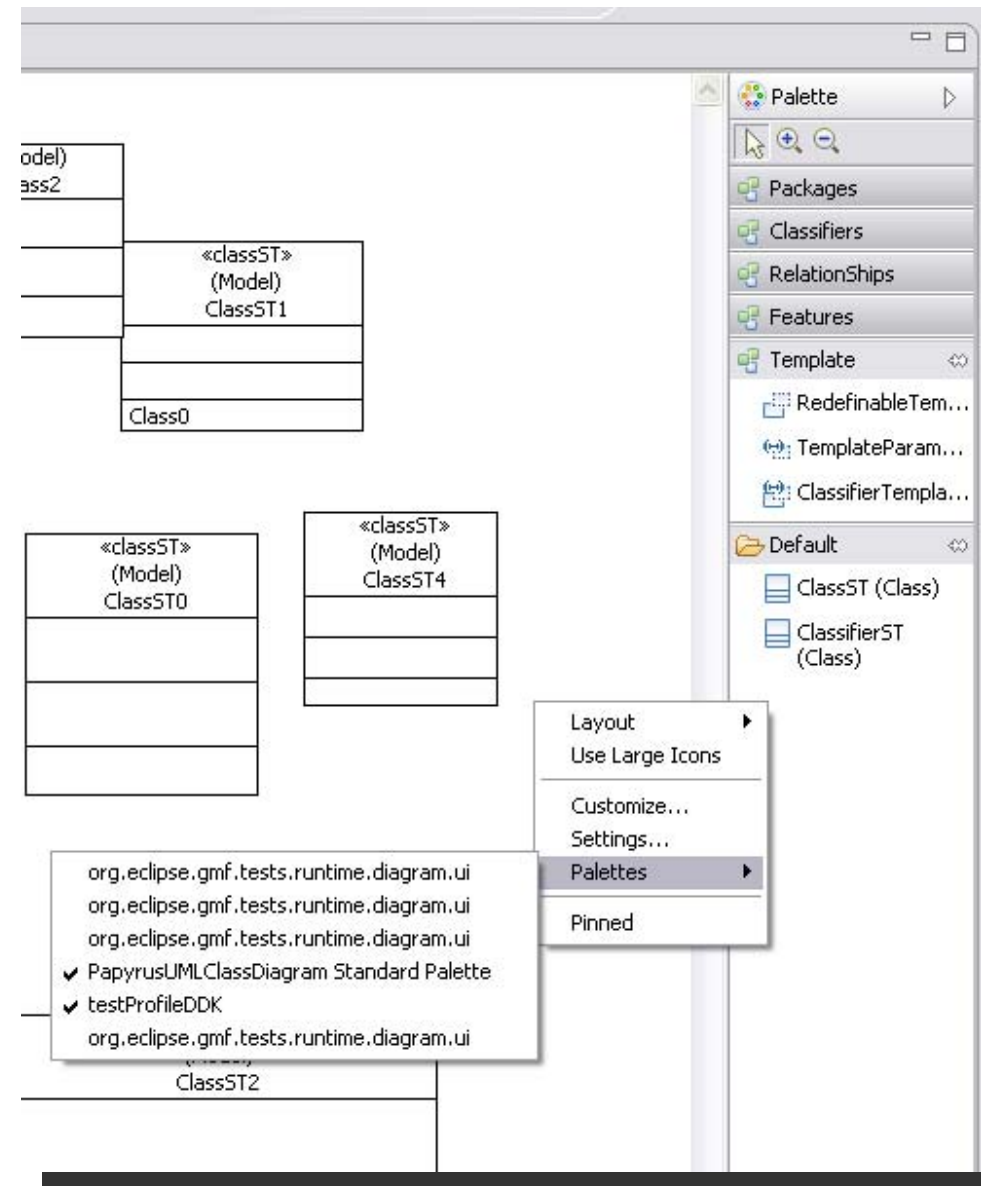
The screenshot illustrates the configuration of a UML profile in Eclipse. It is divided into three main sections:

- Model Explorer (Left):** Shows a project named 'MyRequirementLanguage' containing a profile package. The package includes a 'Class' stereotype, a 'Dependency' stereotype, and two extension stereotypes: 'Extension\_MyRequirement' and 'Extension\_Satisfy'. The 'MyRequirement' stereotype is expanded to show its base class 'base\_Class : Class' and two attributes: 'id : String' and 'txt : String'. The 'Satisfy' stereotype is also visible, along with 'UML 2.2 Metamodel' and 'UML Primitive Types'.
- Class Diagram (Center):** Displays the UML class diagram. It shows two metaclass boxes: '«metaclass» (uml) Class' and '«metaclass» (uml) Dependency'. Below them are two stereotype boxes: '«Stereotype» MyRequirement' and '«Stereotype» Satisfy'. The 'MyRequirement' stereotype has two attributes: '+ id : String [1]' and '+ txt : String [1]'. Arrows indicate that the 'MyRequirement' stereotype is associated with the 'Class' metaclass and the 'Satisfy' stereotype is associated with the 'Dependency' metaclass.
- Property Editor (Bottom):** Shows the configuration for the '«Stereotype» MyRequirement' stereotype. The 'Name' is 'MyRequirement' and the 'Visibility' is 'public'. The 'Is Abstract' property is set to 'false'. Under the 'Features' section, the 'Attributes' list contains three entries: '<Property> id : String', '<Property> txt : String', and '<Property> base\_Class : Class'. There are also controls for adding and removing attributes and operations.



## • Defining specific creation tools

- The palette is yours!
  - Flexibility and functionalities
- Support runtime and predefined customization
  - User friendly customization dialog
- Add creation tools that manage
  - Stereotype application
  - Appearance default choices
  - Model property value on creation
- Mask unused tools
- Mix predefined tools with yours







- **Adapting the model explorer**

- Based on Modisco (Eclipse project)
  - <http://www.eclipse.org/Modisco/>

- Support runtime and predefined customization of the Papyrus model explorer

- **Provides :**

- Query support (Java, OCL)
- Advanced filters
- Configurable look
- Model facets
  - (virtual metamodel extension)
- Tree arrangement
  - (containment, shortcuts)

Query selection dialog

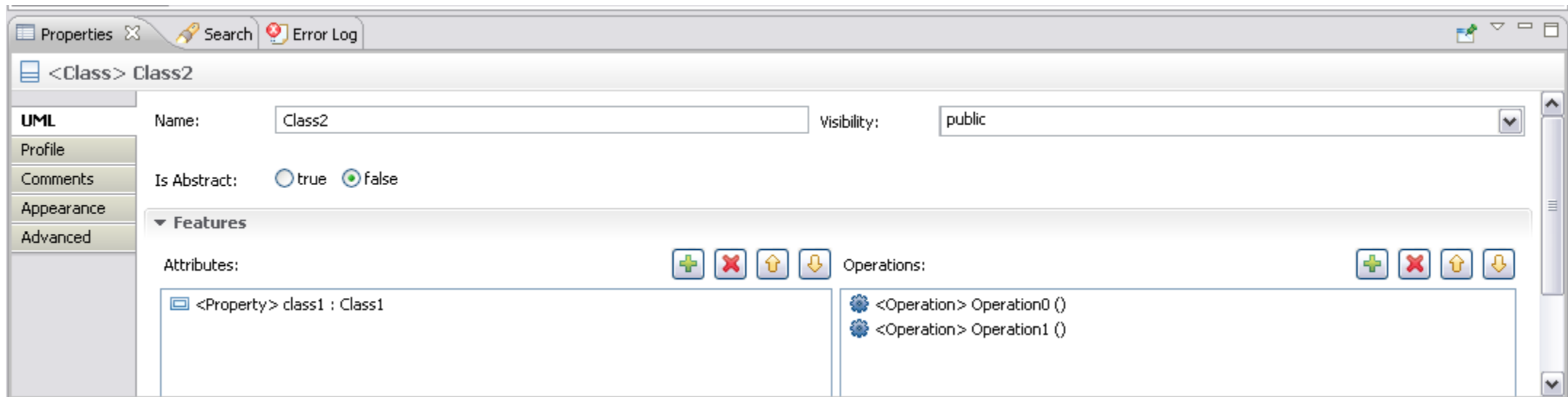
Query result visualization

The screenshot displays the Eclipse IDE interface with several windows open. The 'Papyrus Model Explorer' window shows a tree view of a model with elements like 'MyRequirement0' and 'NewDiagram'. The 'Query Execution' dialog is open, showing a 'Context' of '[Model] <Model>', a 'Query' of 'type filter text', and a list of results including 'PapyrusBrowserQuery', 'MyRequirementElement', and 'isMyRequirementModelQuery'. The 'Execute' button is visible. The 'Table Viewer' window shows 'Model Query Results (1 elements)' in a table format.

| [Label]                   | [Metaclass]            | /eContainer | [Query Context]       | isAbstract | is    |
|---------------------------|------------------------|-------------|-----------------------|------------|-------|
| [Class] <<myRequirement>> | <Class> MyRequirement0 | Class       | [Model] <Model> model | false      | false |



- **Providing dedicated property views (1/3)**
  - **Form-based editor on the model**
    - Define which properties you want to show for an element



Standard property view for UML::Class



- Providing dedicated property views (2/3)

**Customize property view Wizard**

**Content:**

- [EClass] BehavioralFeature -> Namespace, Feature [org.eclipse...
- [EClass] BehavedClassifier -> Classifier [org.eclipse.uml2.uml...
- [EDatatype] Boolean [boolean]
- [EClass] BroadcastSignalAction -> InvocationAction [org.eclipse...
- [EClass] CallAction -> InvocationAction [org.eclipse.uml2.uml.Ce...
- [EClass] CallBehaviorAction -> CallAction [org.eclipse.uml2.uml.C...
- [EEnum] CallConcurrencyKind
- [EClass] CallEvent -> MessageEvent [org.eclipse.uml2.uml.CallE...
- [EClass] CallOperationAction -> CallAction [org.eclipse.uml2.uml...
- [EClass] CentralBufferNode -> ObjectNode [org.eclipse.uml2.ur...
- [EClass] ChangeEvent -> Event [org.eclipse.uml2.uml.ChangeEv...
- [EClass] Class -> EncapsulatedClassifier, BehavedClassifier [o...
- Section Set for single NameElement: org.eclipse.uml2.uml.Nai...
- Section Set for multiple NameElements: org.eclipse.uml2.uml...
- Section Set for single Class: org.eclipse.uml2.uml.Class [1]
- Section Set for single Class stereotyped EClass: org.eclipse.u...
- [EClass] Classifier -> Namespace, RedefinableElement, Type, Te...

**Configuration:**

- Constraints
  - Object Type Constraint: org.eclipse.uml2.uml.Class
  - Stereotypes Constraint: [Ecore::EClass]
- Content
  - Section: singleClass\_EClassSection in tab: basicTab
    - Fragment: fragment\_single\_Class\_Eclass
      - ExpandableContainerDescriptor
        - GridLayout (2, true)
          - EMFTController for: className
          - EMFTController for: xmlContentKind

**Preview:**

< Back   Next >   Finish   Cancel



- Providing dedicated property views (3/3)

The screenshot shows the Eclipse IDE interface with the Properties view open for a UML Class named `Class2`. The view is titled "<Class> Class2". The left sidebar contains tabs for UML, Profile, Comments, Appearance, and Advanced. The main area displays the following properties:

- Name:** `Class2`
- Visibility:** `public`
- Is Abstract:**  true  false
- Features:**
  - Attributes:**  <Property> class1 : Class1
  - Operations:**
    - <Operation> Operation0 ()
    - <Operation> Operation1 ()

The screenshot shows the Eclipse IDE interface with the Properties view open for an EClass named `Class0`. The view is titled "<<eClass>> <Class> Class0". The left sidebar contains tabs for UML, Profile, Comments, Appearance, and Advanced. The main area displays the following properties:

- Name:** `class0`
- Visibility:** `protected`
- EClass:**
  - Class Name:** `Serialization`
  - XML Co... kind:**  <Unset>  Unspecified  Empty  Simple  Mixed  ElementOnly



## • Registering a new diagram

- No runtime configuration tool yet...
- The diagram is registered via a specific Eclipse extension point
  - The diagram content is defined by inheriting from existing diagram (Class here)
  - Behavior and element aspect can be modified
- Papyrus SysML diagrams are created by extending UML2 diagrams

The screenshot shows the Eclipse IDE's 'Extensions' view for the 'org.eclipse.papyrus.requirement.diagram' plug-in. The view is divided into two main sections: 'All Extensions' and 'Extension Element Details'.

**All Extensions:** This section displays a list of extensions for the plug-in. The list includes:

- org.eclipse.gmf.runtime.diagram.ui.paletteProviders
- org.eclipse.papyrus.diagram.common.paletteDefinitio
- org.eclipse.ui.menus
- org.eclipse.ui.commands
- org.eclipse.ui.handlers
- org.eclipse.papyrus.core.papyrusDiagram
  - (editorDiagram)
  - My Requirement Diagram (creationCommand)** (highlighted)
- org.eclipse.gmf.runtime.diagram.core.viewProviders
- org.eclipse.gmf.runtime.diagram.ui.editpartProviders
- org.eclipse.ui.preferencePages

Buttons for 'Add...', 'Remove', 'Up', and 'Down' are visible next to the list.

**Extension Element Details:** This section shows the properties of the selected extension element, 'creationCommand'. The properties are:

- id\*:** org.eclipse.papyrus.requiremen (Browse...)
- label\*:** My Requirement Diagram
- creationCommandClass\*:** org.eclipse.papyrus.requiremen (Browse...)
- language\*:** requirement (Browse...)
- icon:** icons/RequirementDiagram.gif (Browse...)
- creationCondition:** (Browse...)

The bottom of the screenshot shows the Eclipse IDE's navigation bar with tabs for Overview, Dependencies, Runtime, Extensions, Extension Points, Build, MANIFEST.MF, plugin.xml, and build.properties.

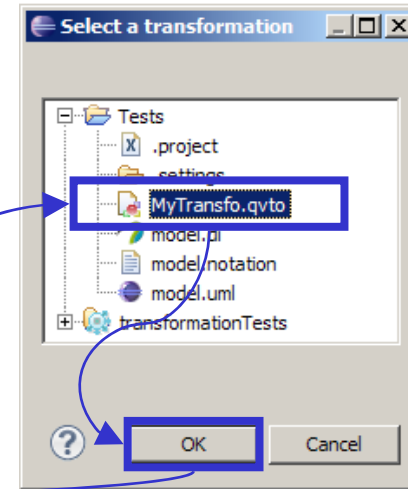
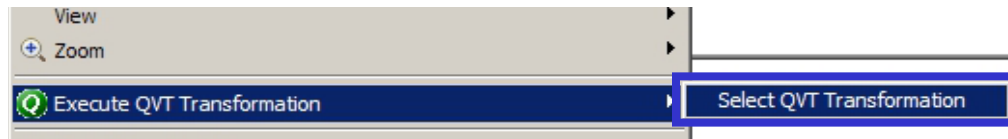


# Execution of QVTo transformations in Papyrus (seq.)

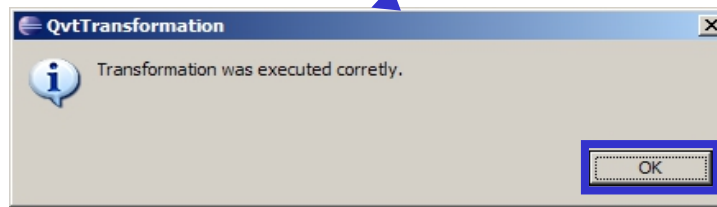
## • Launching the transformation

### ▪ Scenario:

- Within the editor, right-click and select following action:

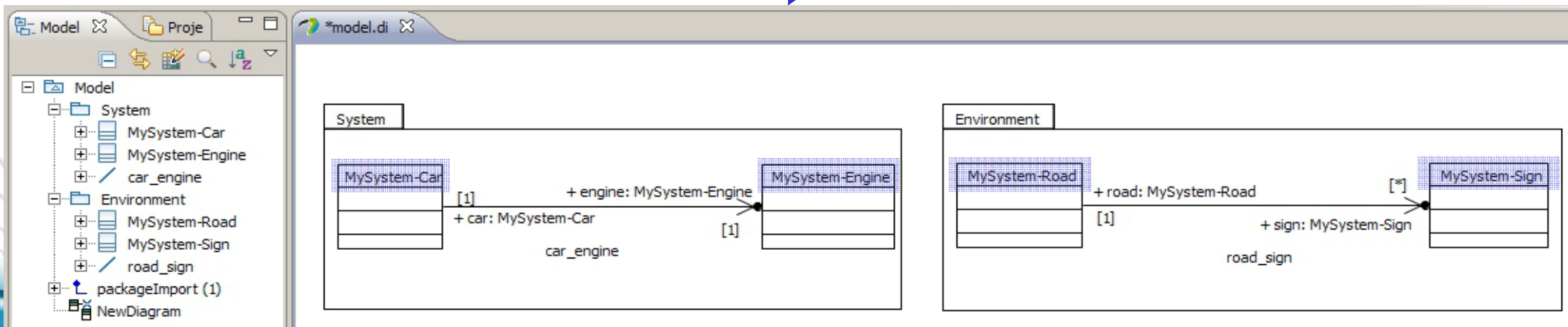


## • Message to confirm (or an error dialog with diagnostic if it failed):



Classes have been renamed according to the model transformation definition.

## • Resulting model:





# Summary on model transformation within Papyrus

## • Benefits

- The transformation is executed using Papyrus' editingDomain.
- Therefore, the transformation is considered as a regular command:



Modifications performed through the model transformation can be undone/redone!

## • Current limits

- Transformations signature:
  - Transformations must have only one INOUT parameter,
  - And the metamodel for this parameter must be UML.
- If the transformation has OUT and other IN parameters, run directly with QVTo
  - ➔ The transformation cannot be undone and redone!

## • Future work

- Overcome the limits aforementioned.
- Select a transformation from the Model Explorer.



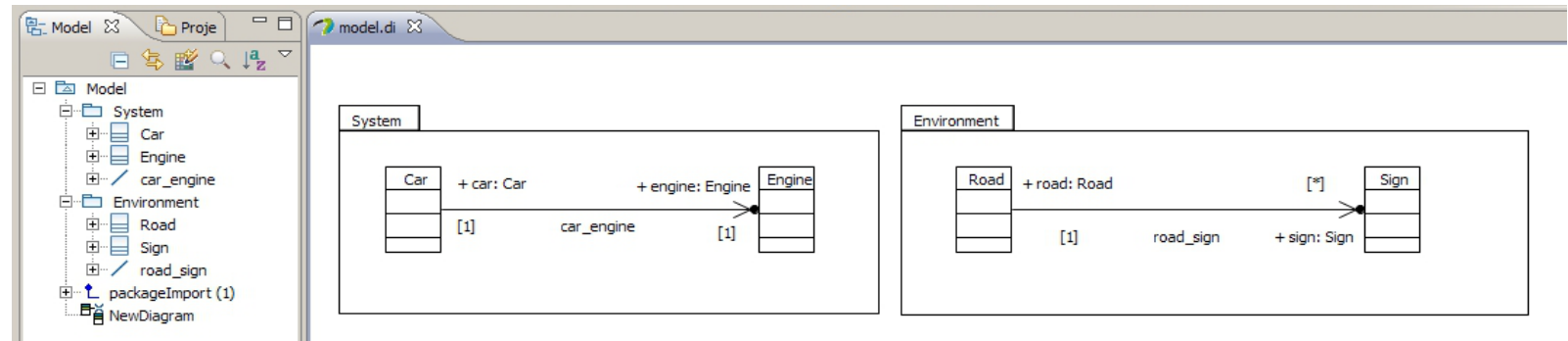
# Execution of QVTo transformations in Papyrus

- **A three steps process :**

- Develop transformations in QVTo editor,
- Select a transformation to execute during modeling,
- Done!

- **Let's try it...**

- Initial model



- Example of a model transformation in QVTo

```
MyTransfo.qvto
modeltype UML uses 'http://www.eclipse.org/uml2/3.0.0/UML';
transformation MyTransformation(inout model : UML) {

    main() {
        //Get the root of the model
        var root := model.rootObjects()![UML::Model];
        //Get all the classes in this model
        var classes := model.objectsOfType(Class)->asSequence();

        //Prefix all of them
        classes->forEach(e1) {
            e1.name := "MySystem-"+e1.name;
        }
    }
}
```





- **First release -> 0.7.0 (Mid-July)**
- **Next steps: next release 0.7.1 (Early october)**
  - **Current main activities focused on stability**
    - Intensive test, validation and debugging phase!
  - **Improve current customization facilities**
    - ... and complete with user-friendly tool configuration (Papyrus DSL workbench)
  - **Extending language support (EAST-ADL2, MARTE)**
  - **Usability improvements**
  - **Integrate side-components (code generators...)**
  - **Enable diff of models**

**➔ Contributions and feedback welcomed!**



- **For developers...**
  - [http://wiki.eclipse.org/Papyrus\\_Developer\\_Guide](http://wiki.eclipse.org/Papyrus_Developer_Guide)
  - <http://dev.eclipse.org/mailman/listinfo/mdt-papyrus.dev>
- **For vendors/consumers...**
  - <http://www.eclipse.org/papyrus>
- **For users...**
  - <news://news.eclipse.org/eclipse.papyrus>
- **Papyrus project lead contact: [sebastien.gerard@cea.fr](mailto:sebastien.gerard@cea.fr)**