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

Acknowledgments

- **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?

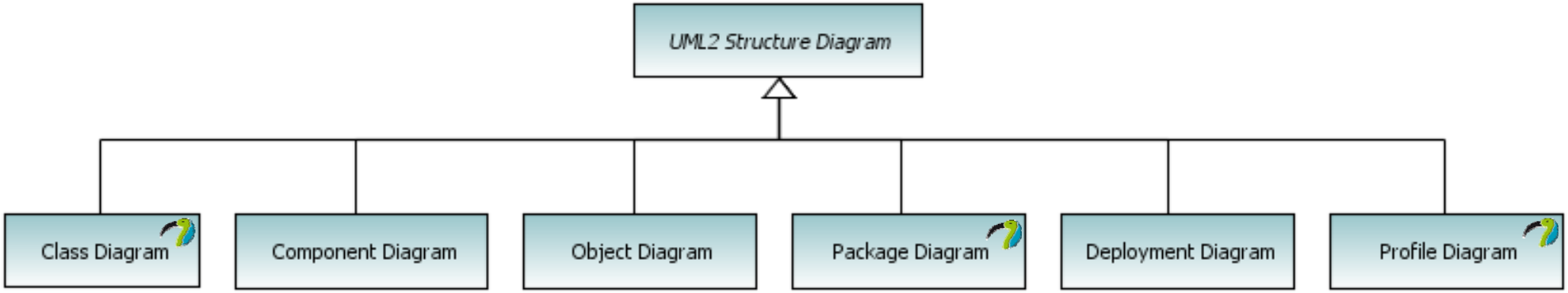
Standards have traditionally provided major boosts to technological progress !

- **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

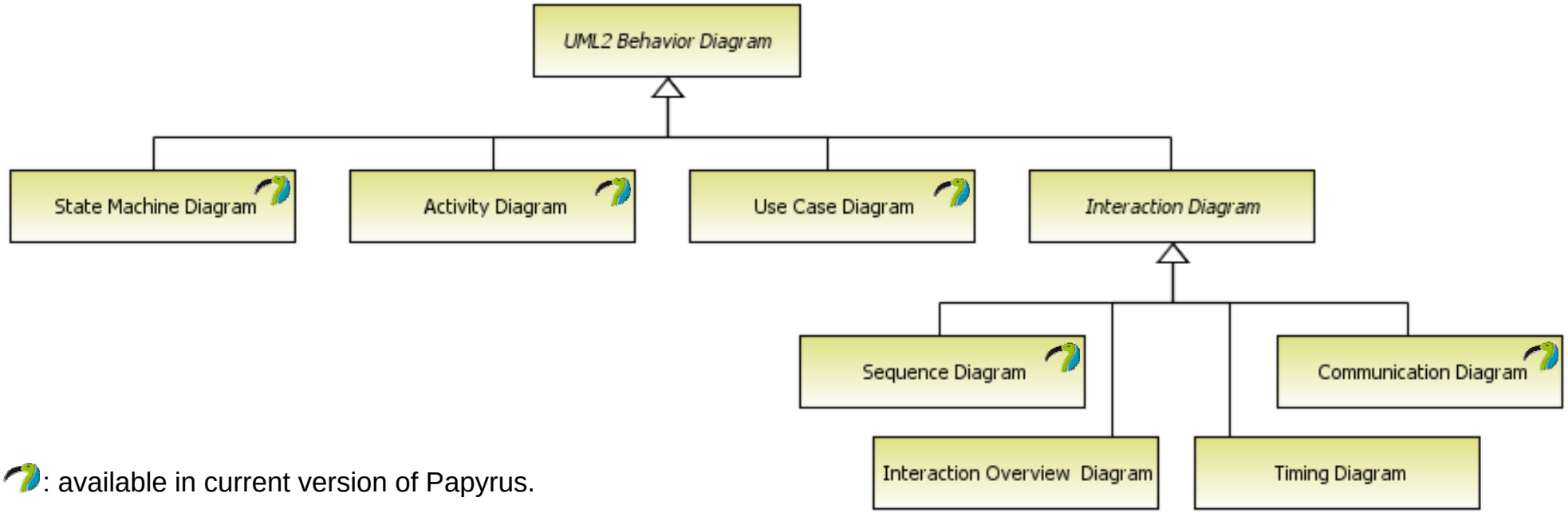
- **The Object Management Group (OMG) has created the Model-Driven Architecture initiative:**
  - A comprehensive set of standards in support of MBE including standard modeling languages: **UML2, MARTE and SysML.**

UML2, a familly of modeling languages

- **6 diagram kinds for structure modeling**



- **7 diagram kinds for behavior modeling**

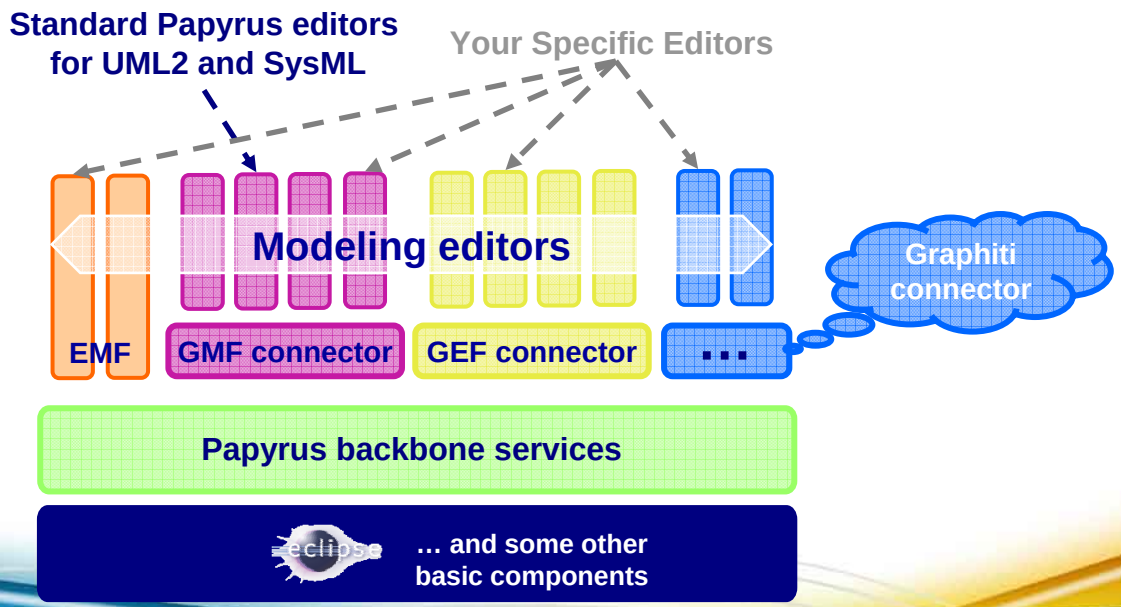


: available in current version of Papyrus.

Quick architecture tour

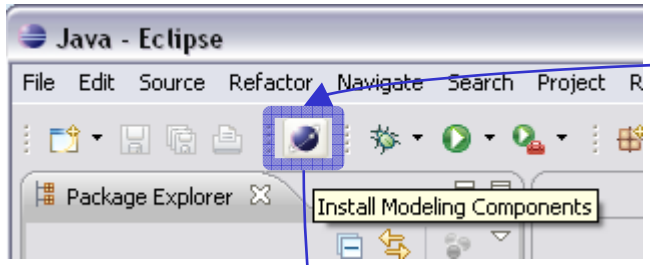
- **Eclipse based as usual…**
  - Based on well known Eclipse modeling components
    - EMF, GMF, UML2, Modisco, xtext…
    - UML2 and SysML diagram based in GMF (custom generation)

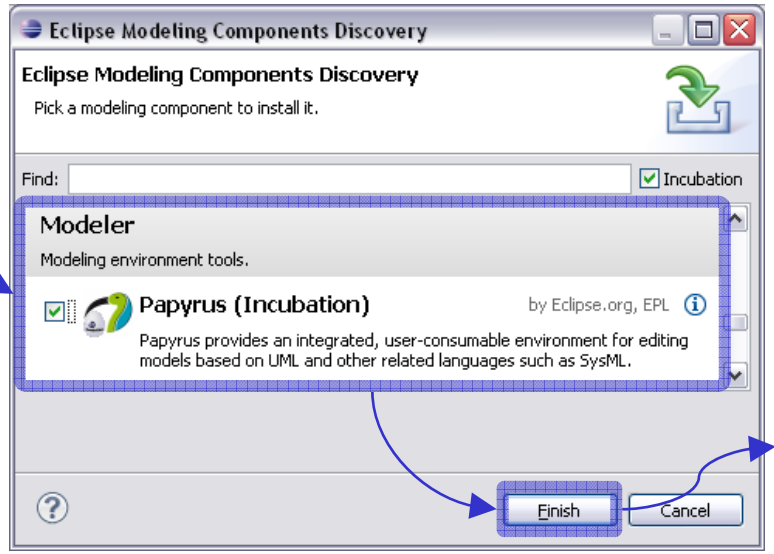
- **… built as an integration platform for diagrams**
  - Supporting various modeling languages
    - Not necessary UML2
  - Graphical or text-based editors
  - Supporting several frameworks
    - GMF, GEF ready (connector available)
    - Extensible (dedicated connector) to future frameworks (Graphiti)

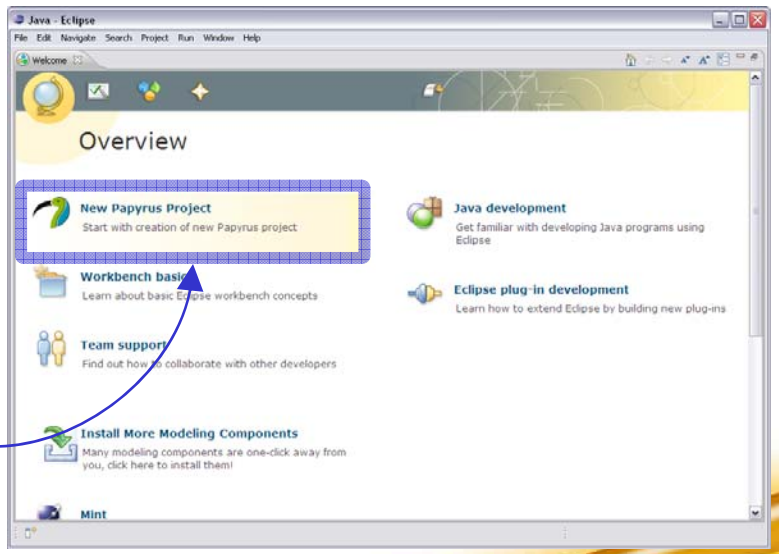


Papyrus standard download

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

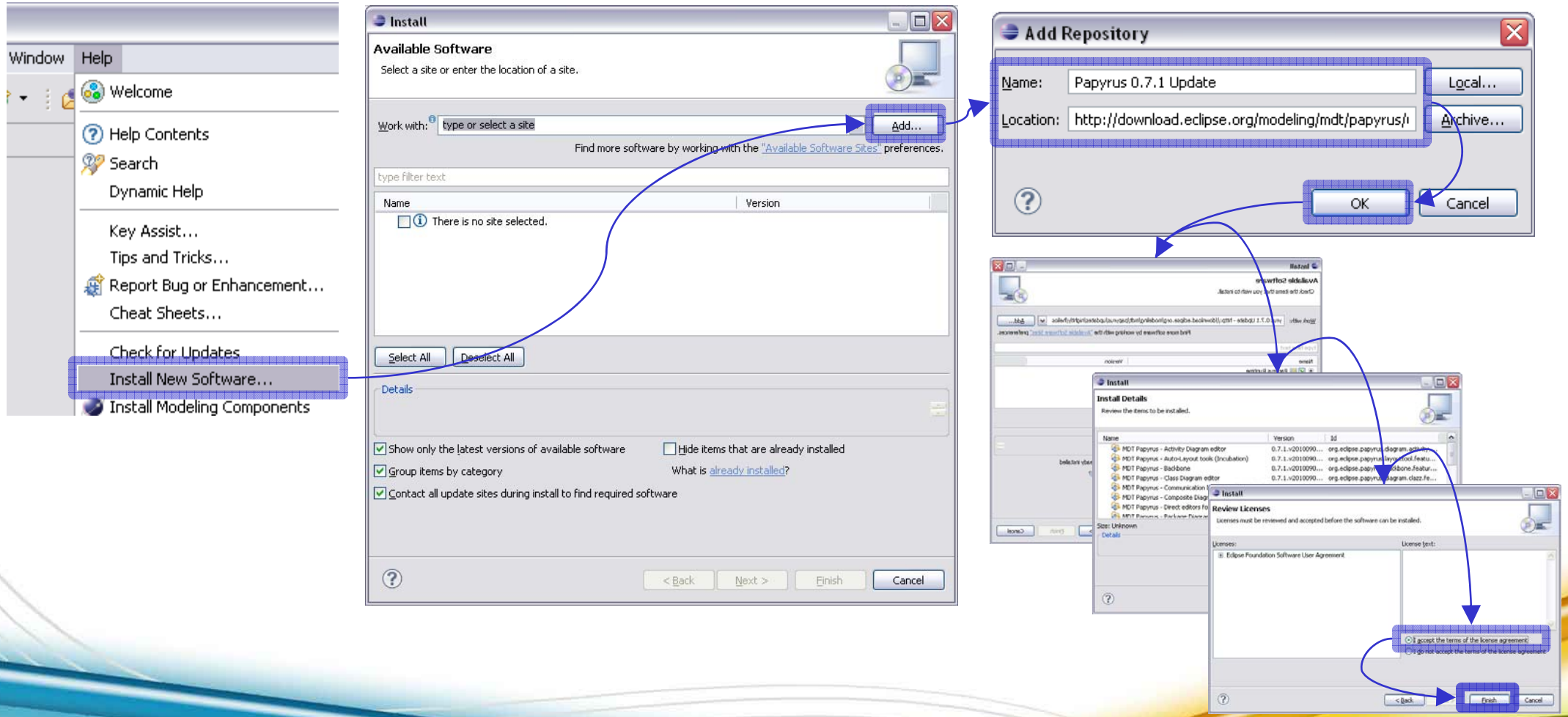






Papyrus nightly build version download

- **Via the Papyrus update site**
  - Scenario:
    - Download the Eclipse Modeling Platform (www.eclipse.org/downloads),
    - Unzip the downloaded file and start Eclipse.exe,
    - In the menu bar, check Help > Install New Software…



Papyrus web site: www.eclipse.org/papyrus

Papyrus - Mozilla Firefox

Fichier Édition Affichage Historique Marque-pages Outils ?

http://www.eclipse.or

Papyrus

Home Downloads Users Members Committers Resources Projects About Us

Download
Eclipse Distribution, Update Site, Drop-ins

Support
Bug Tracker, Newsgroup, Professional Support

Documentation
Tutorials, Examples, Videos, Online Reference

Getting Involved
CVS, Workspace Setup, Wiki, Committers

Papyrus

Current Status

Headlines in the web

UML2

DSL

SysML

Text In Papyrus

And much more...

Papyrus Update Sites - Mozilla Firefox

http://www.eclipse.org/modeling/mdt/papyrus/updates/index.php

Papyrus Update Sites

Home Downloads Users Members Committers Resources Projects About Us

Papyrus »
- Download
- Update Sites
- Documentation
- Support
- Getting Involved

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.

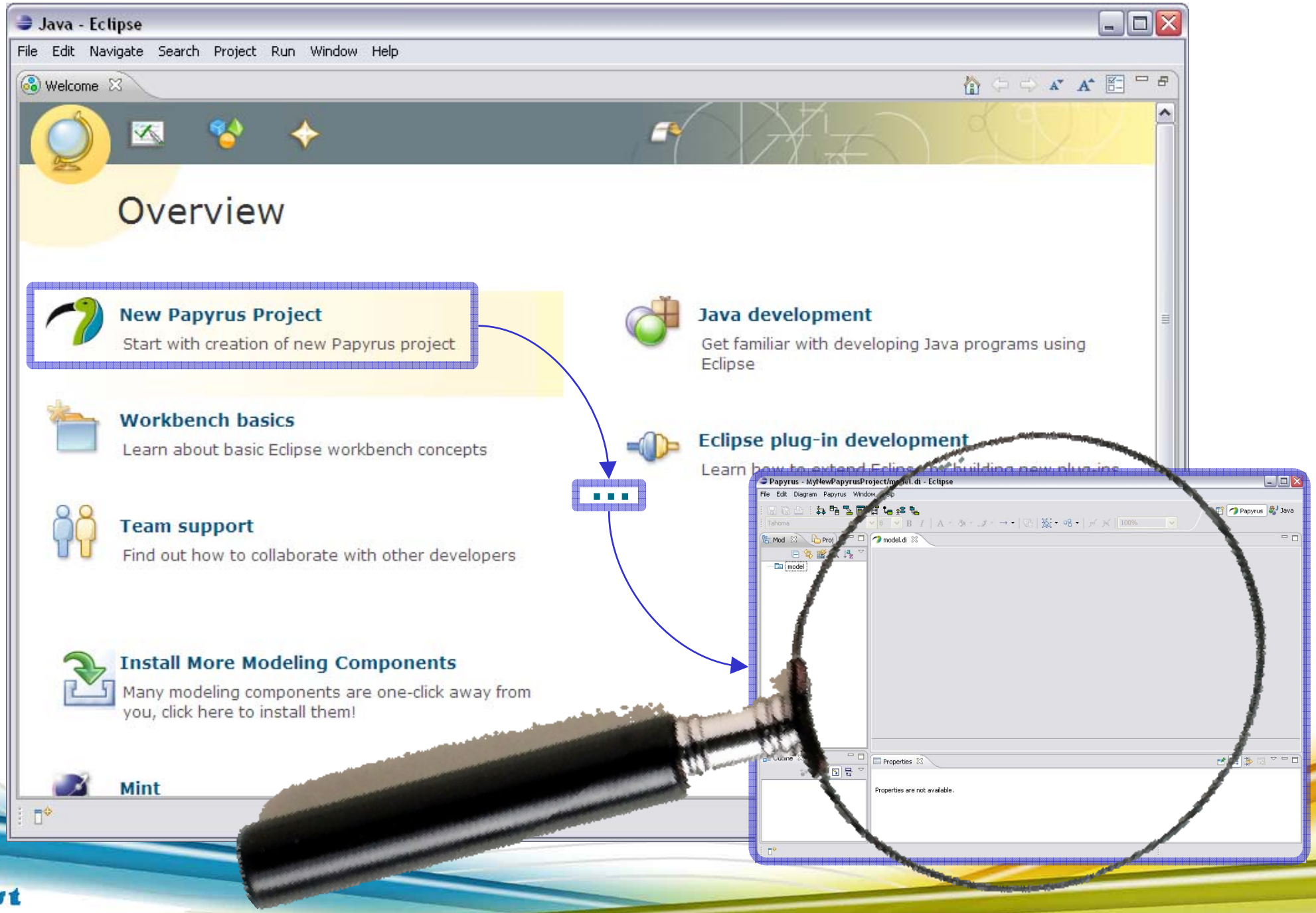
Main update site: (Recommended)

- **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)

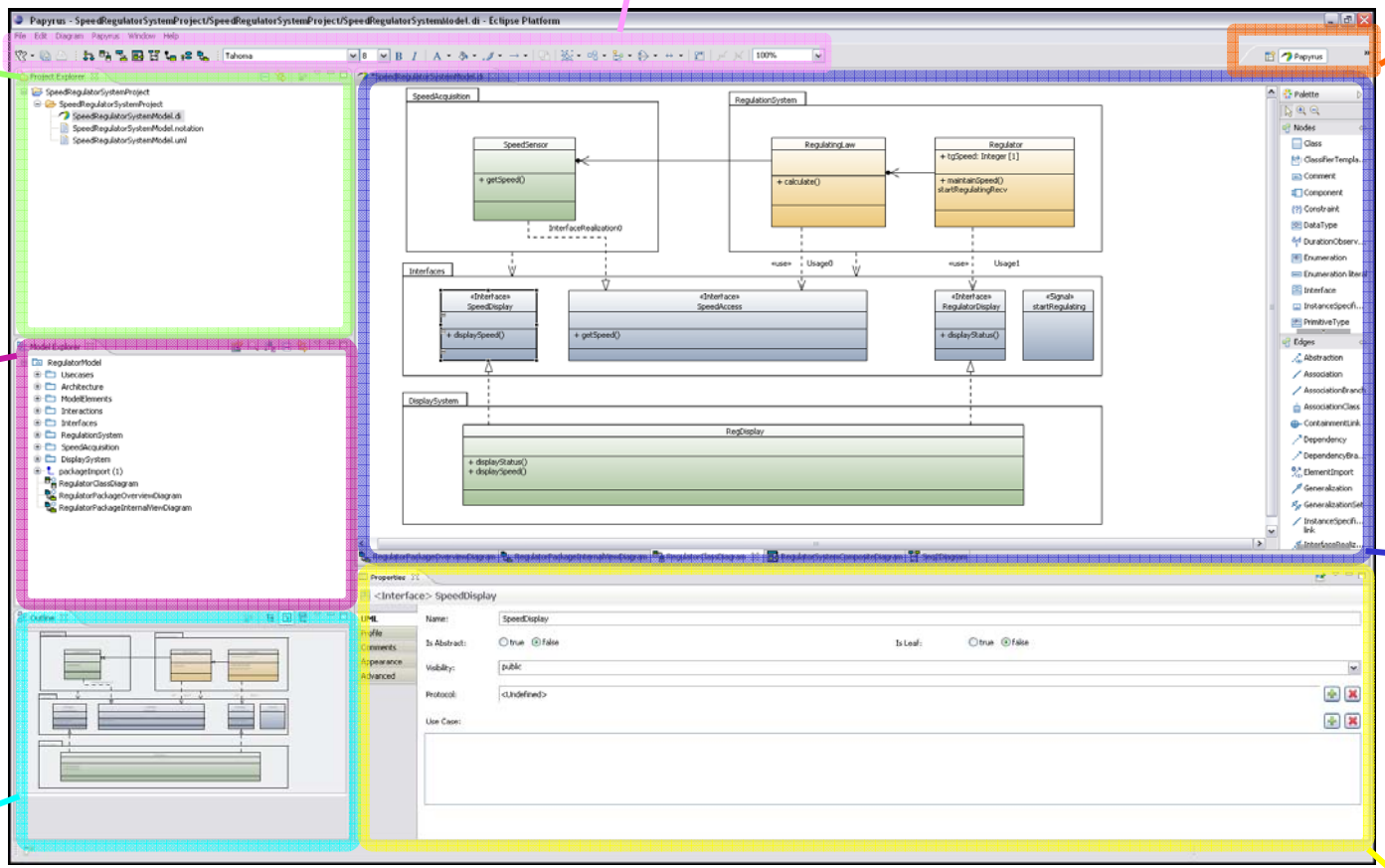
Papyrus wellcome page



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 /hyde, …



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).

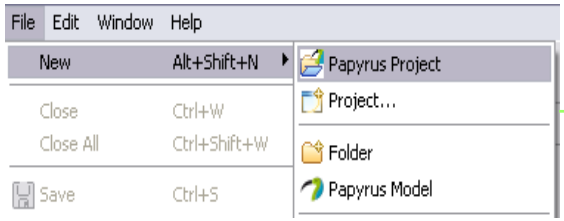
Property view: form-based model editor enabling to view & edit model element properties.

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

Project creation

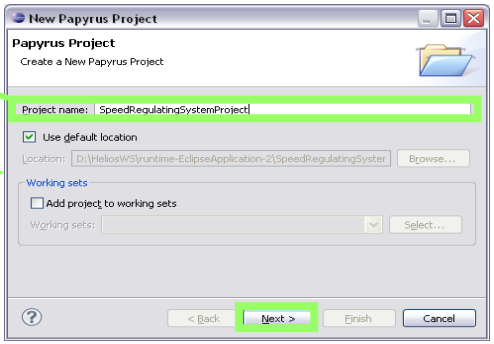
- **Creating a Papyrus project**
  - In the Menu bar, click on: File > New>Papyrus Project

1



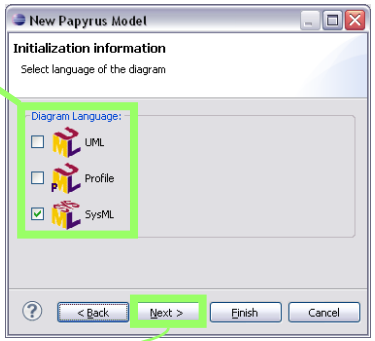
2

Enter a project name and press Next.

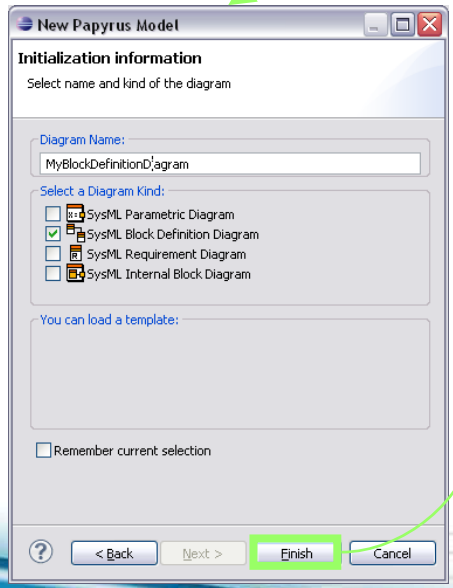


3

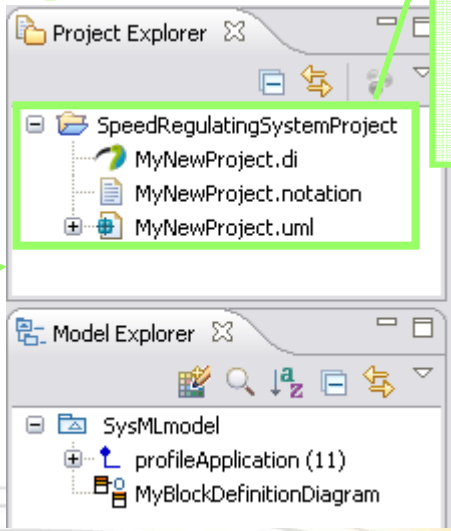
Select a modeling language (e.g. SysML).



3



4

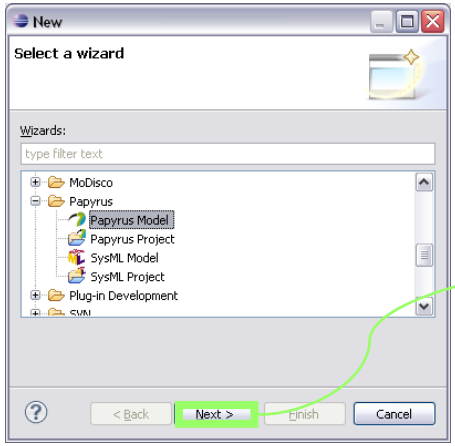


New created UML model:
- ***.di: tool metadata**
  - ➥ **entry point for starting modeling!**
- ***.notation: graphical data**
- ***.uml: UML model data**

Model management

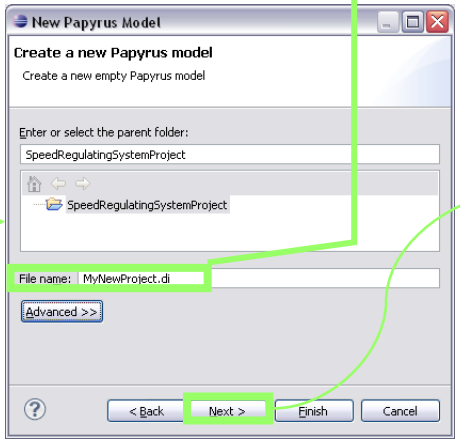
- **Creating a new Papyrus model**
  - Within the project explorer view:
    - Select a project > Right click on it > New > Other

1

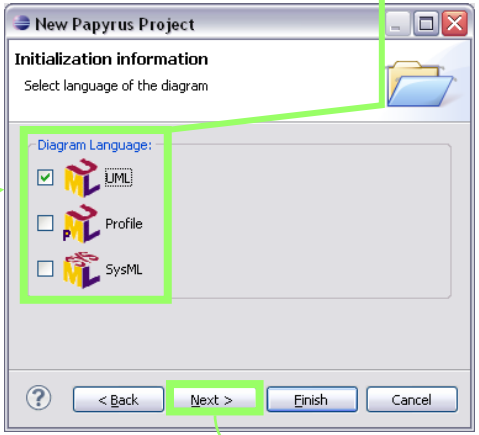


Enter a model name and press Next.

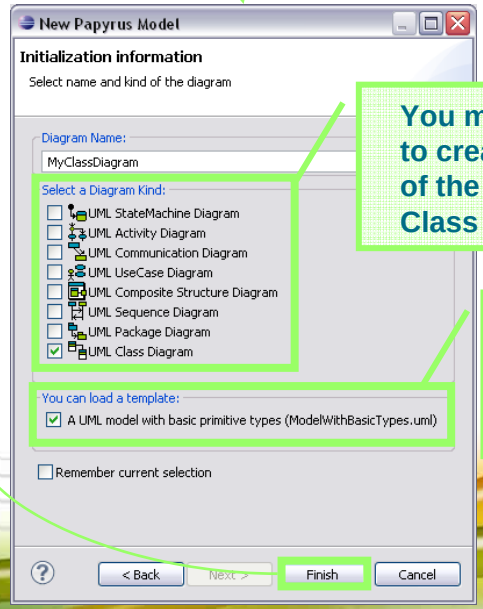
2



Select a modeling language (e.g. UML)



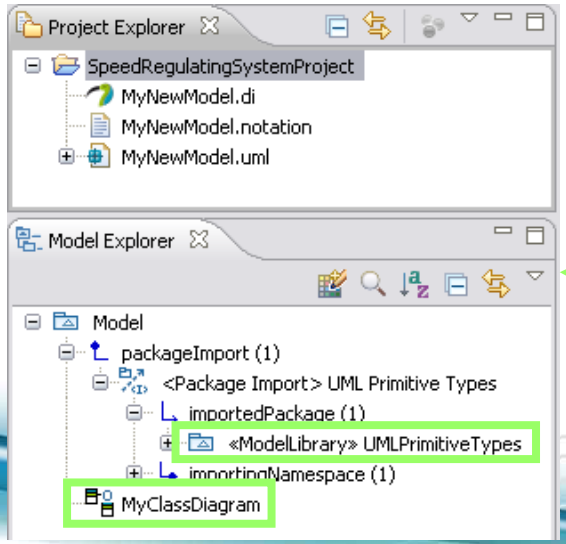
4



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

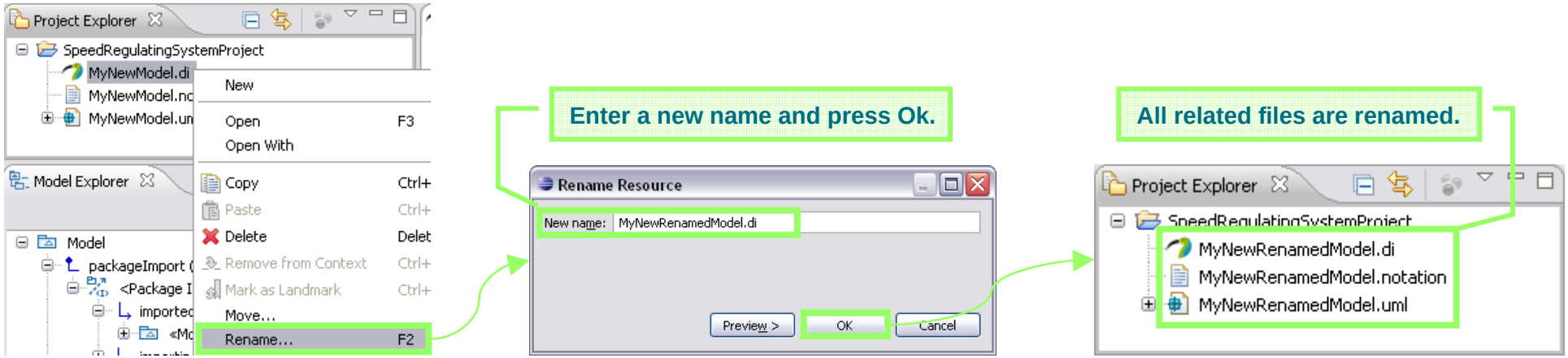
You may also select model template (e.g. Model Template importing UML Basic data type).

5



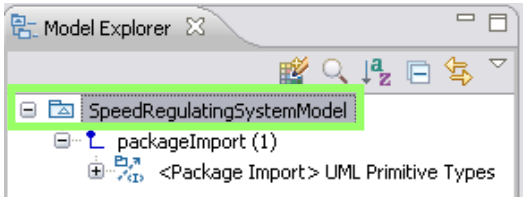
Model File Renaming

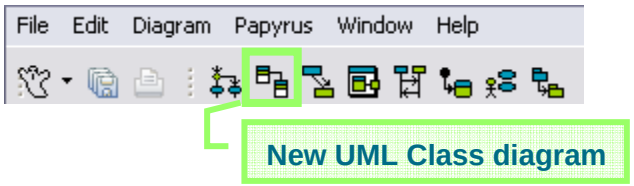
- **Within the project explorer:**
  - Select the model to rename
  - Right-click on it > Rename (short cut ➔ F2)

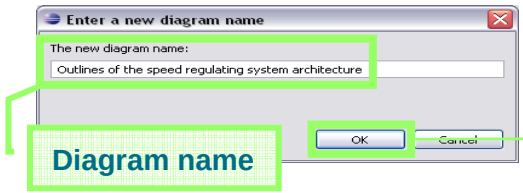


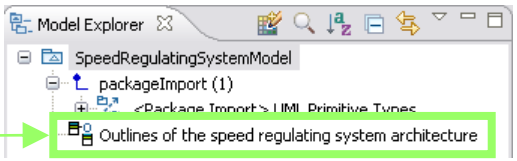
Diagrams creation

- **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.

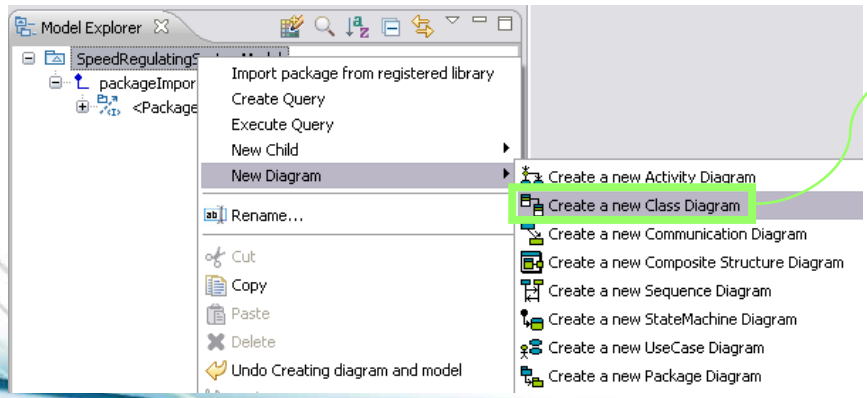


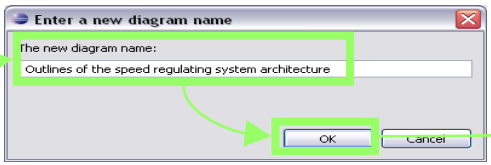


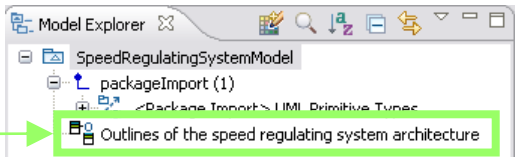




    - Scenario 2: left-click on the selected element > New Diagram > Create a new Class Diagram

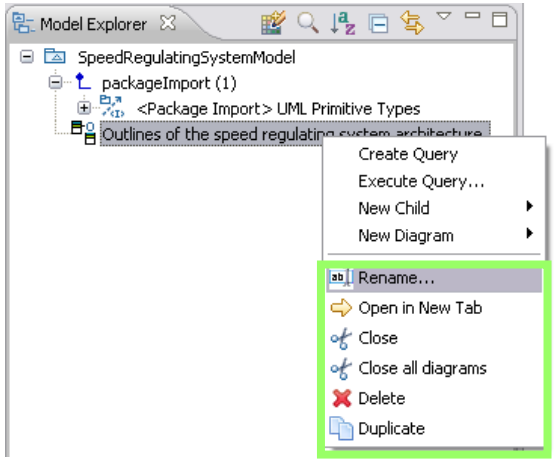




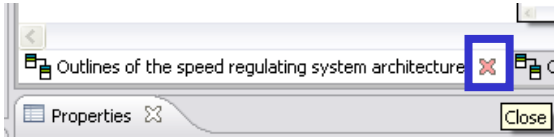


Diagrams lifecycle management

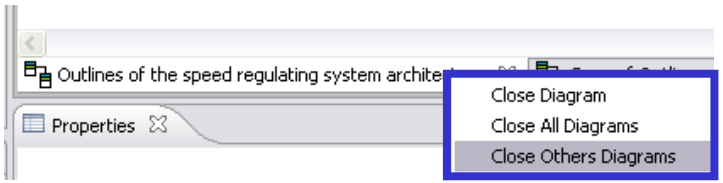
- **Diagrams can be:**
  - Renamed
  - Closed
  - Open in a new tab
  - Deleted
  - Duplicated
  - Moved from holder to a new one in the model explorer
- **Scenario 1: right-click on it in the model explorer > select a command.**



- **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.

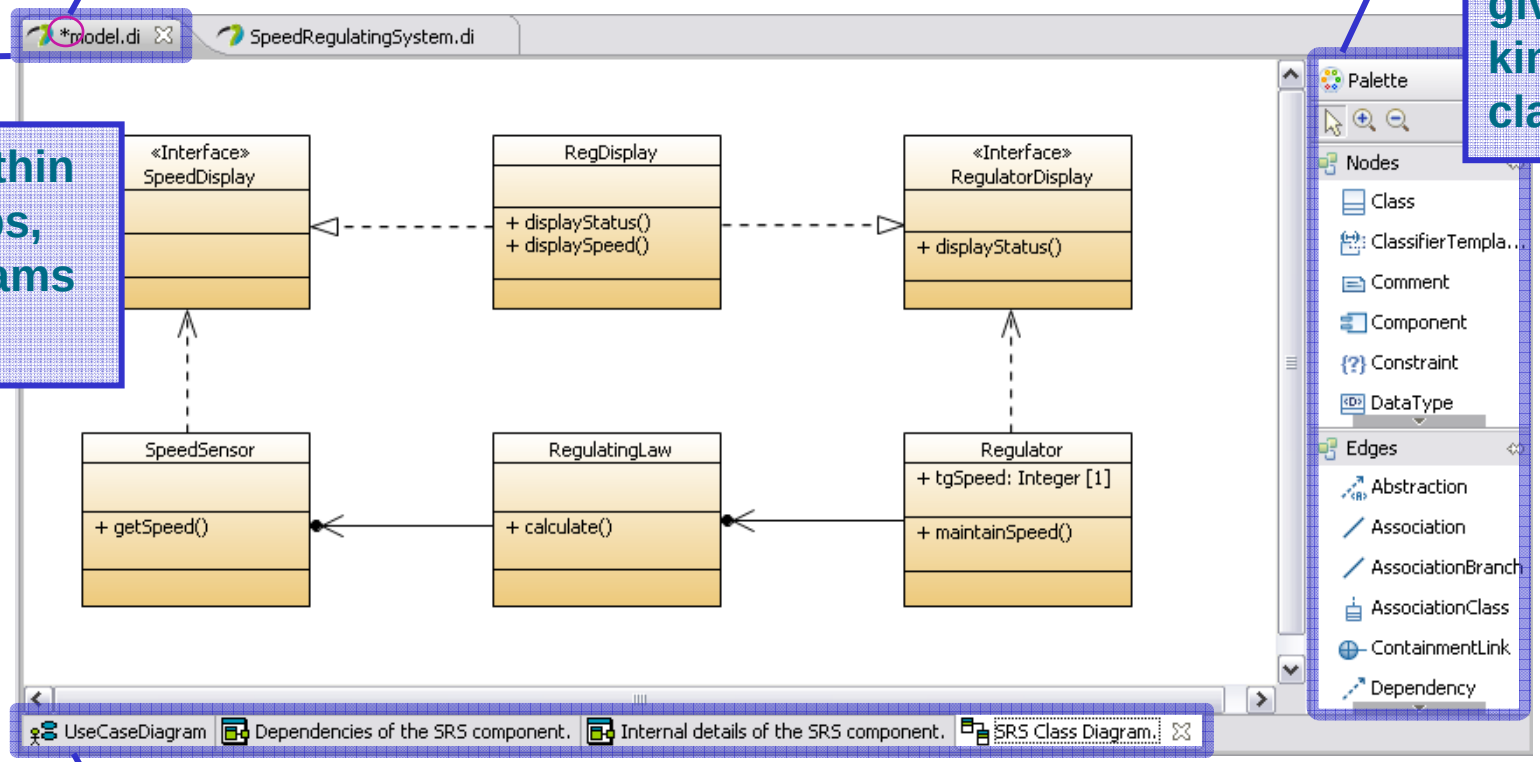
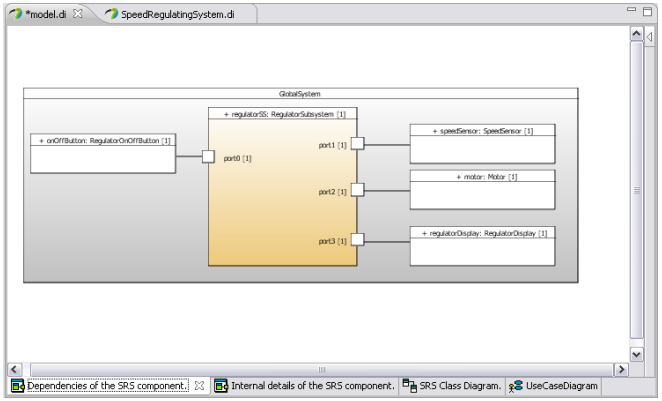
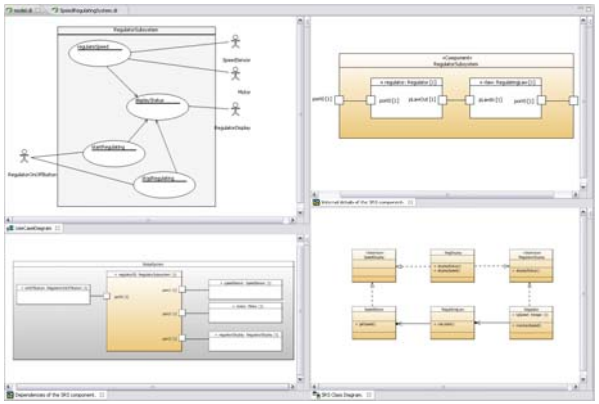


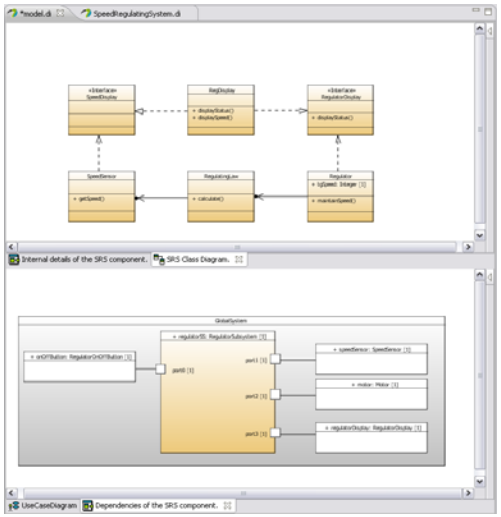
Diagram tabs: each tab holds for one open diagram (e.g., 4 diagrams are open).

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.





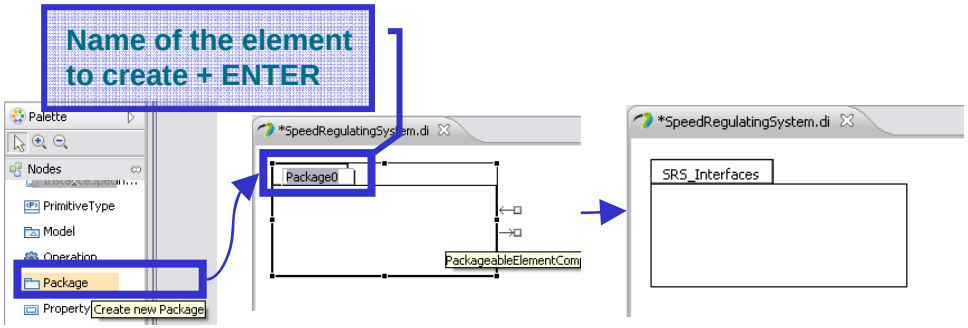


- **Graphical editors are made of two element kinds**
  - Nodes
    - E.g., Class, Lifeline, State.
  - Edges
    - Associations, Message, Transitions.

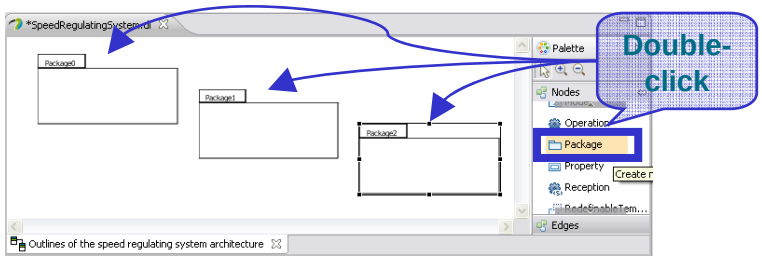
Now, let's see how to populate a diagram…

Creating nodes

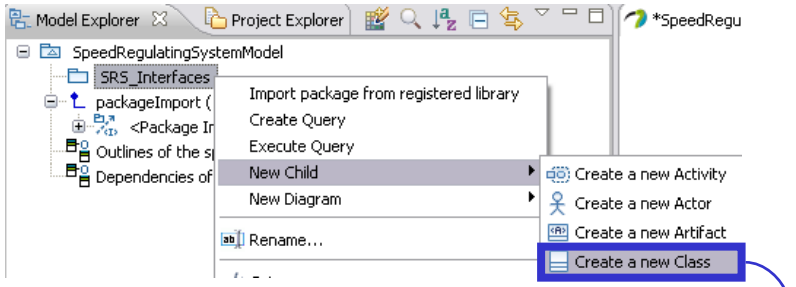
- **Using the palette**
  - Scenario 1:
    - Wihtin the palette, click the kind of element to create.
    - Click within the diaram editor frame where you want to create the model element.
    - Enter a name and press Return.

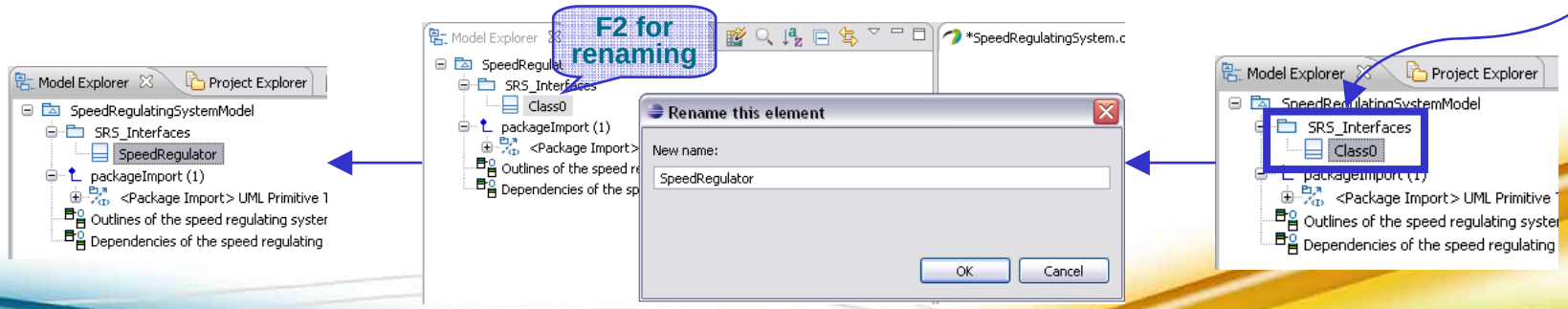


  - Scenario 2 (for creating several model elements):
    - Wihtin the palette, double-click the kind of element to create as many time as you want to create model elements.



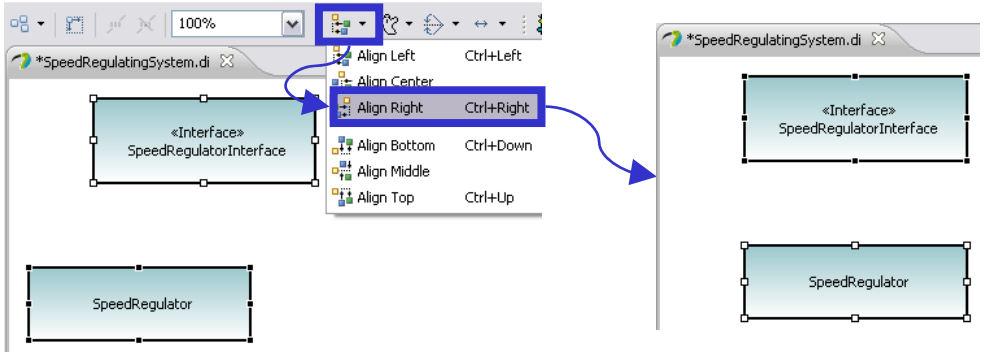
- **Using the model explorer**
  - Scenario 1:
    - Within the model explorer, right-click on the model element that will contain the element to create.
    - Select New Child and then select the kind of model element to create.
    - To rename the created element, select it and either press F2 or right-click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





Graphical alignements 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 alignement policy.

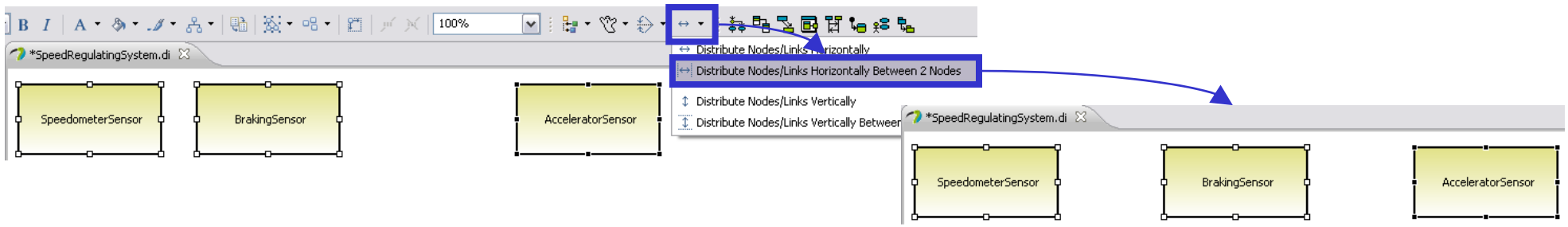


It is the last selected element that is used as reference position!

  - Scenario 2:
    - Select the nodes to align,
    - Then, hit keys **Ctrl** + Arrow (←, →, ↑ or ↓).

Graphical distributions of nodes

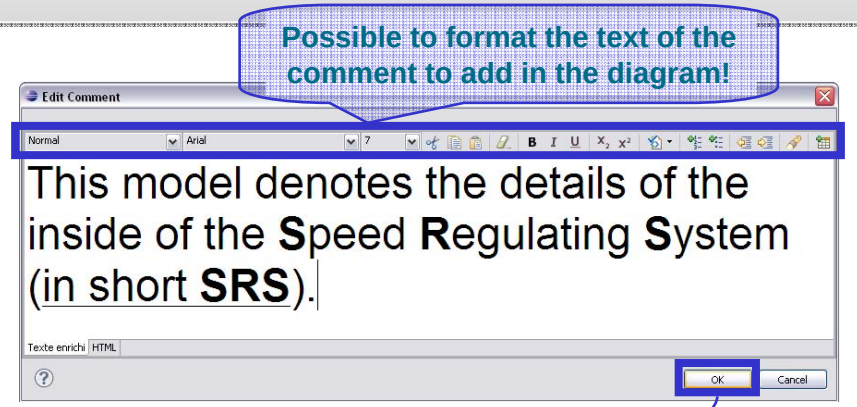
- **Scenario:**
  - Select the nodes to distribute,
  - Apply one of the distribution strategies available from the Papyrus action bar.

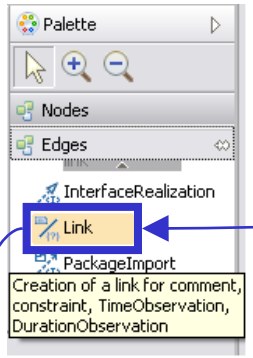


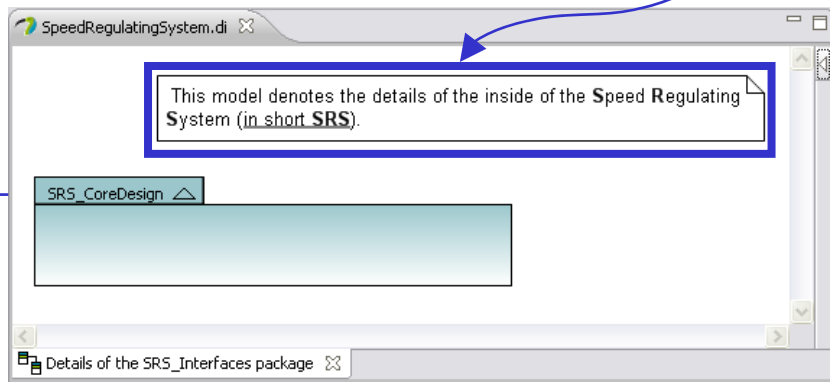
- **Notes**
  - Two kinds of distribution are possible for both horizontal and vertical directions
    - ↔ or ↕ : nodes are distributed between both most external selected nodes.
    - ↕ or ↔ : nodes are distributed in the range of their container
  - Example on ports within the composite class diagram

Adding a Comment

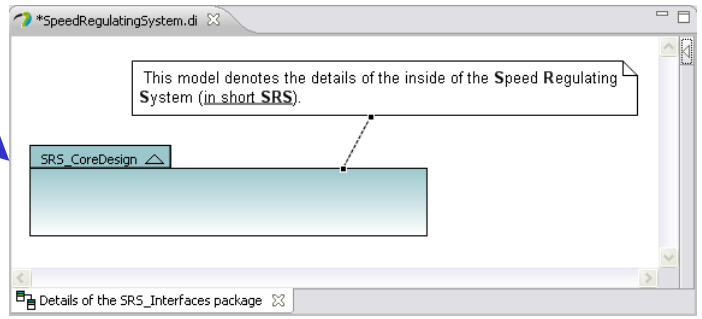
- **Adding a new Comment**
  - Scenario:
    - Add an Comment node on the diagram,
    - Type your comment using the enriched textual editor.
    - Then, add the links between the new Comment and the elements being element.
      - For that purpose, let's use the tool "Link" in the Palette,







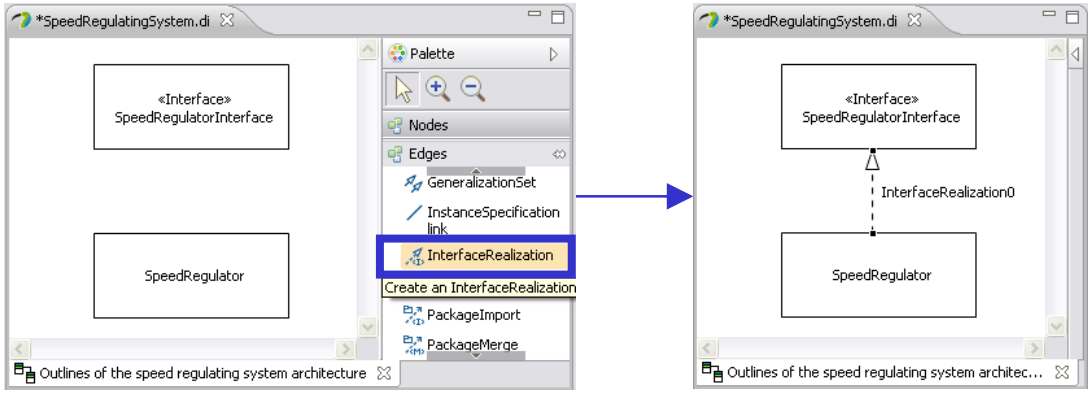
      - And draw a link between the created Comment and each element being commented.



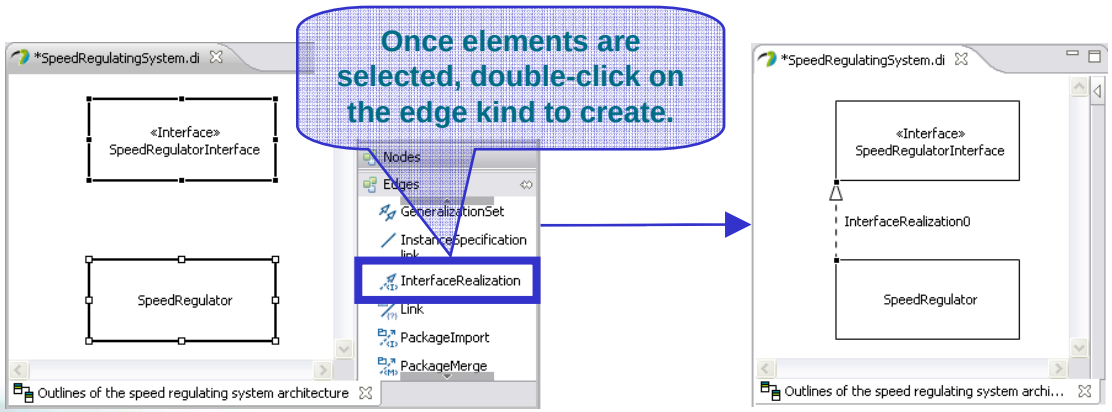
    - **Note:** Any kind of model elements, either nodes or edges, may be commented!

Creating edges

- **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.

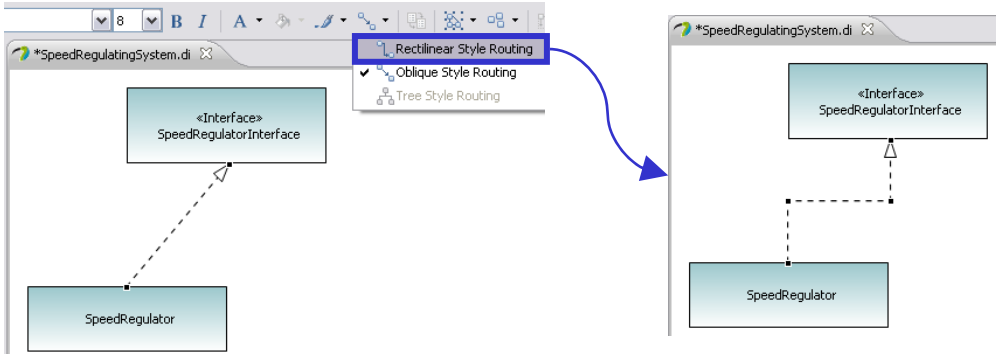


- **Scenario 2:**
  - Within the diagram editor frame, select both source and target elements.
  - Next, within the palette, double-click on the edge kind you want to create.

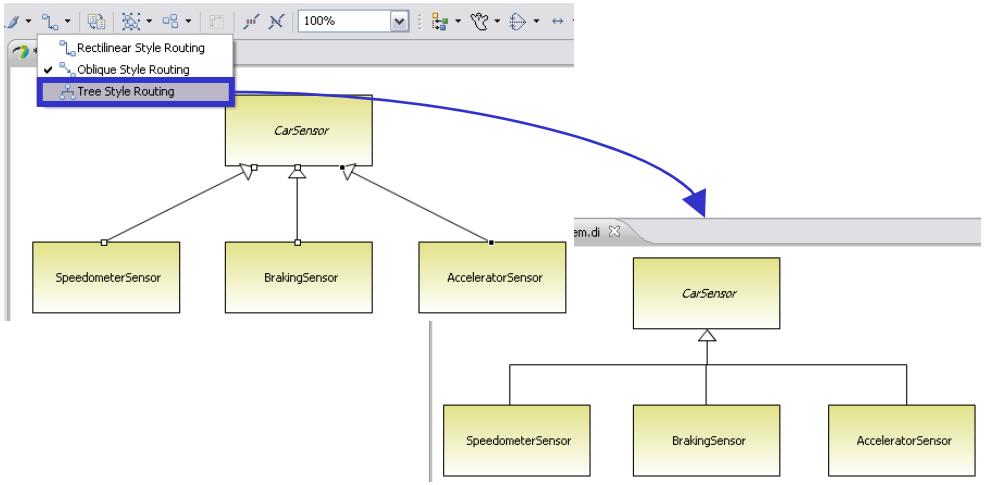


Routing edge policies

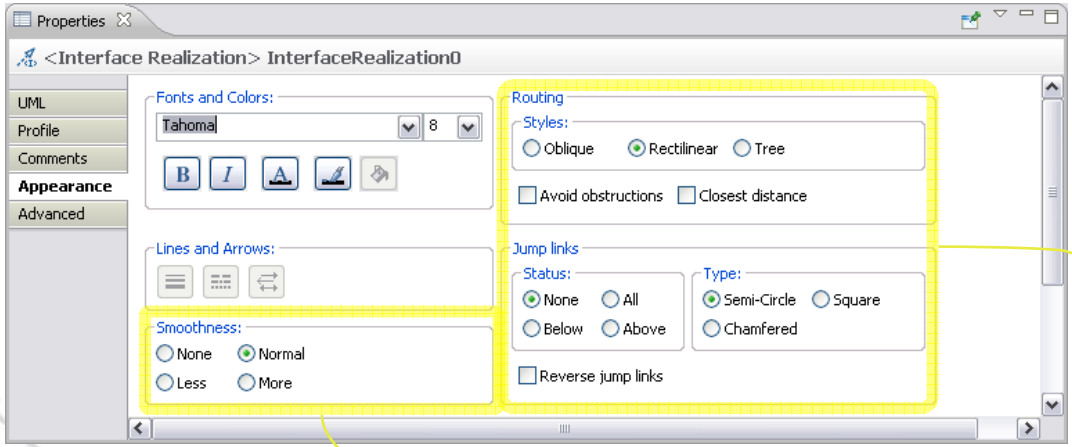
- **Oblique versus rectilinear routing policies for edges**

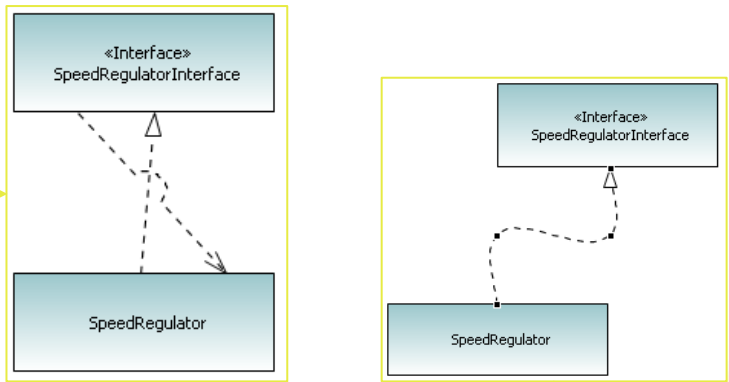


- **Using tree style routing**
  - Scenario:
    - Select the edges to route and apply tree-style routing policy.



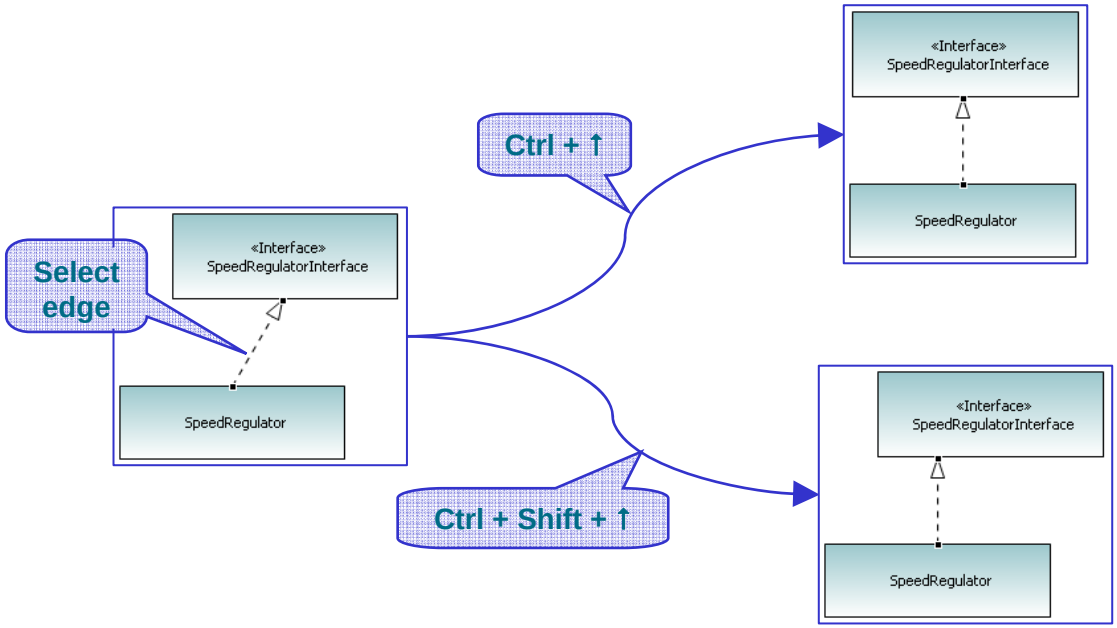
- **Possible parameterizations of routing policies within the Appearance tab of the property view:**

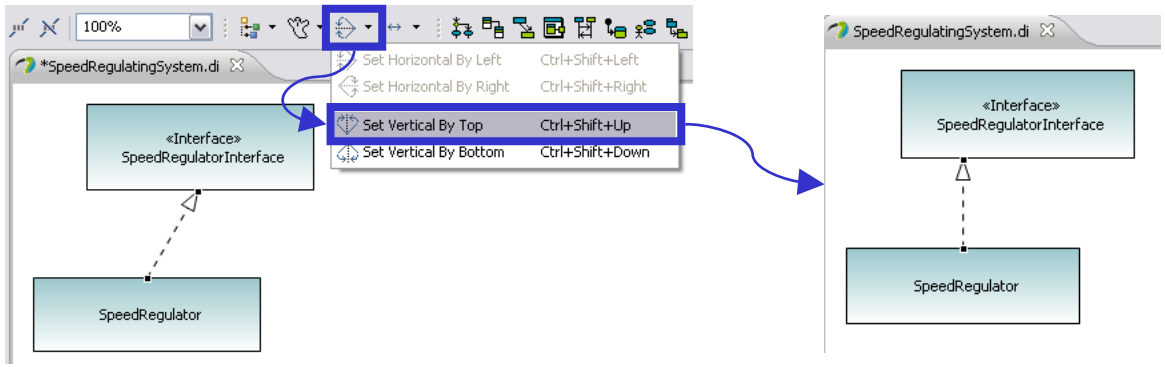




Re-routing Edges

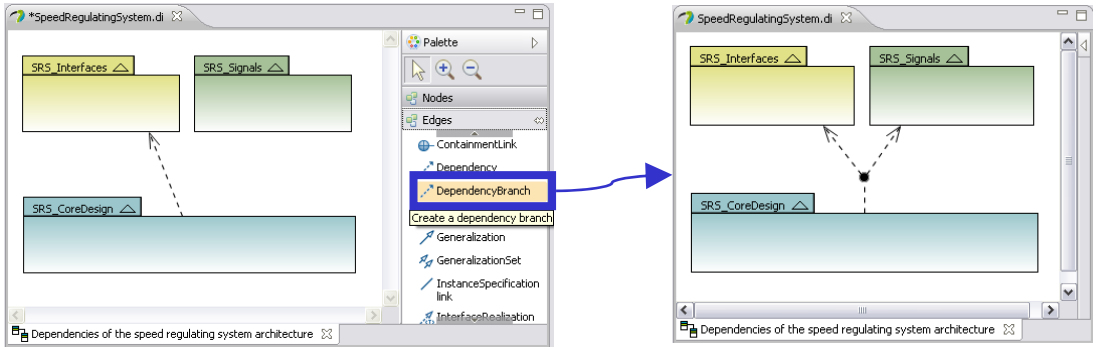
- **Using short keys**
  - Scenario 1:
    - Select the edges to reroute,
    - Hit (**Ctrl** + < ↑, ↓, ← or → >)
      ➔ only opposite nodes move.
  - Scenario 2:
    - Select the edges to reroute,
    - Hit (**Ctrl** + **Shift** + < ↑, ↓, ← or → >)
      ➔ only edge anchors move.
- **Using Papyrus tool bar**
  - Scenario:
    - Select the edges to reroute,
    - Select on the command of the menu .



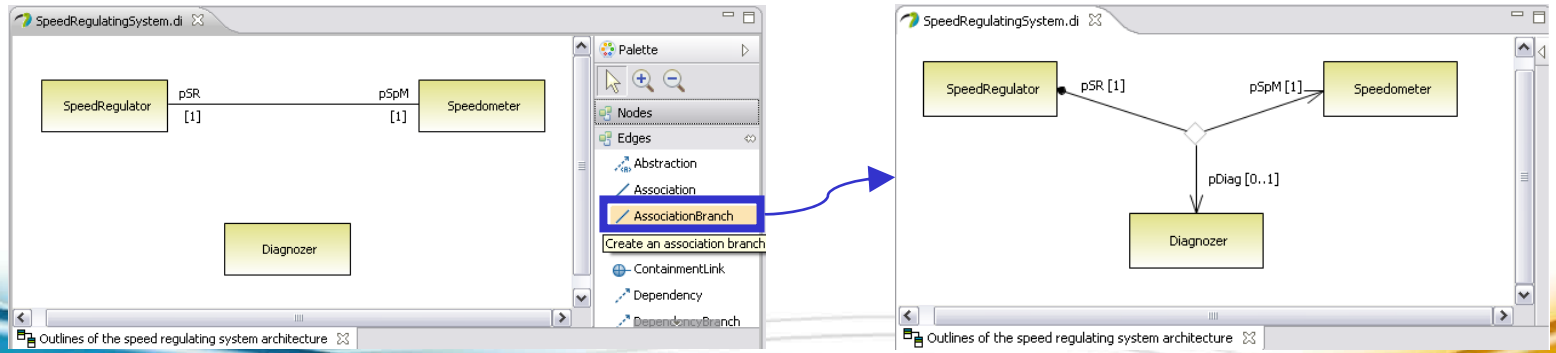


Modeling multi-branches edges

- **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.

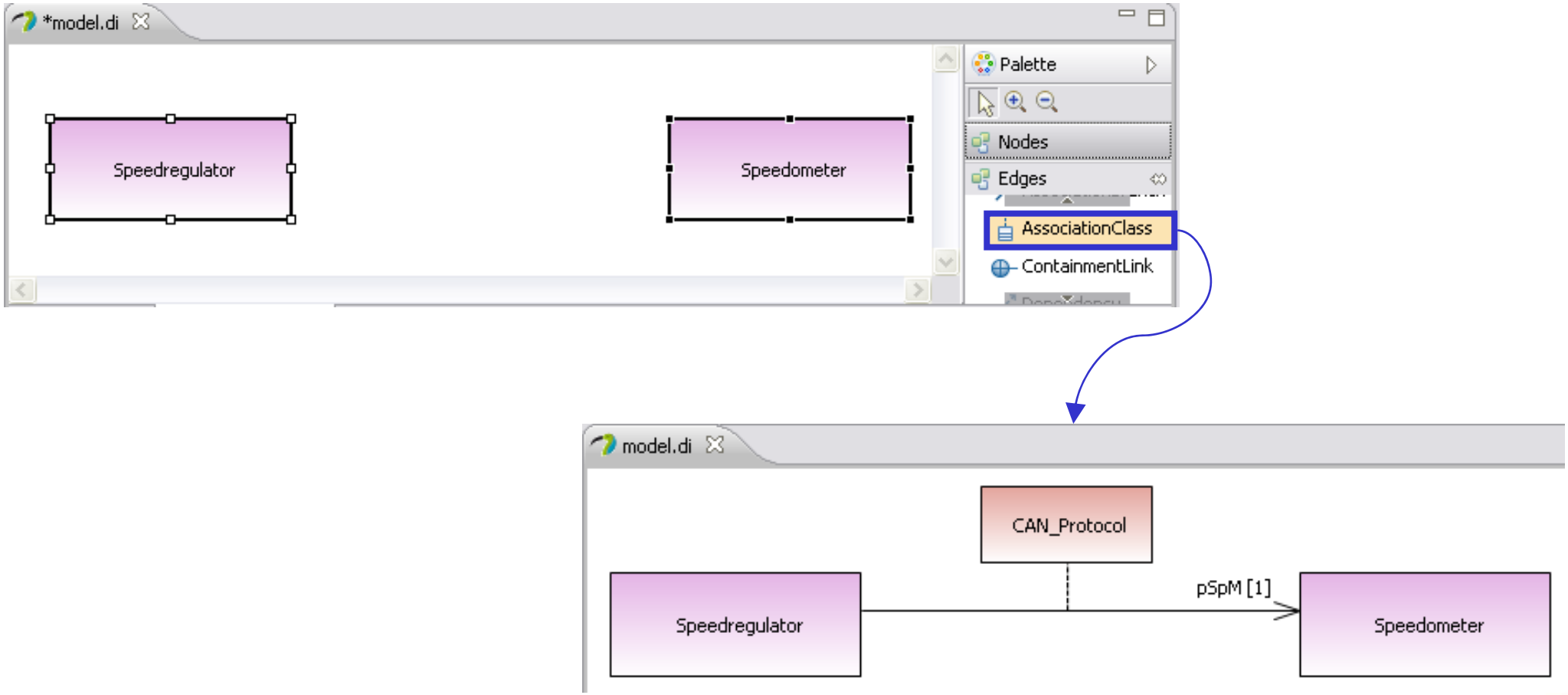


- **Modeling multi-associations**
  - Create an association between two of the elements to
  - Add a branch using the tool “AssociationBranch" in the palette.



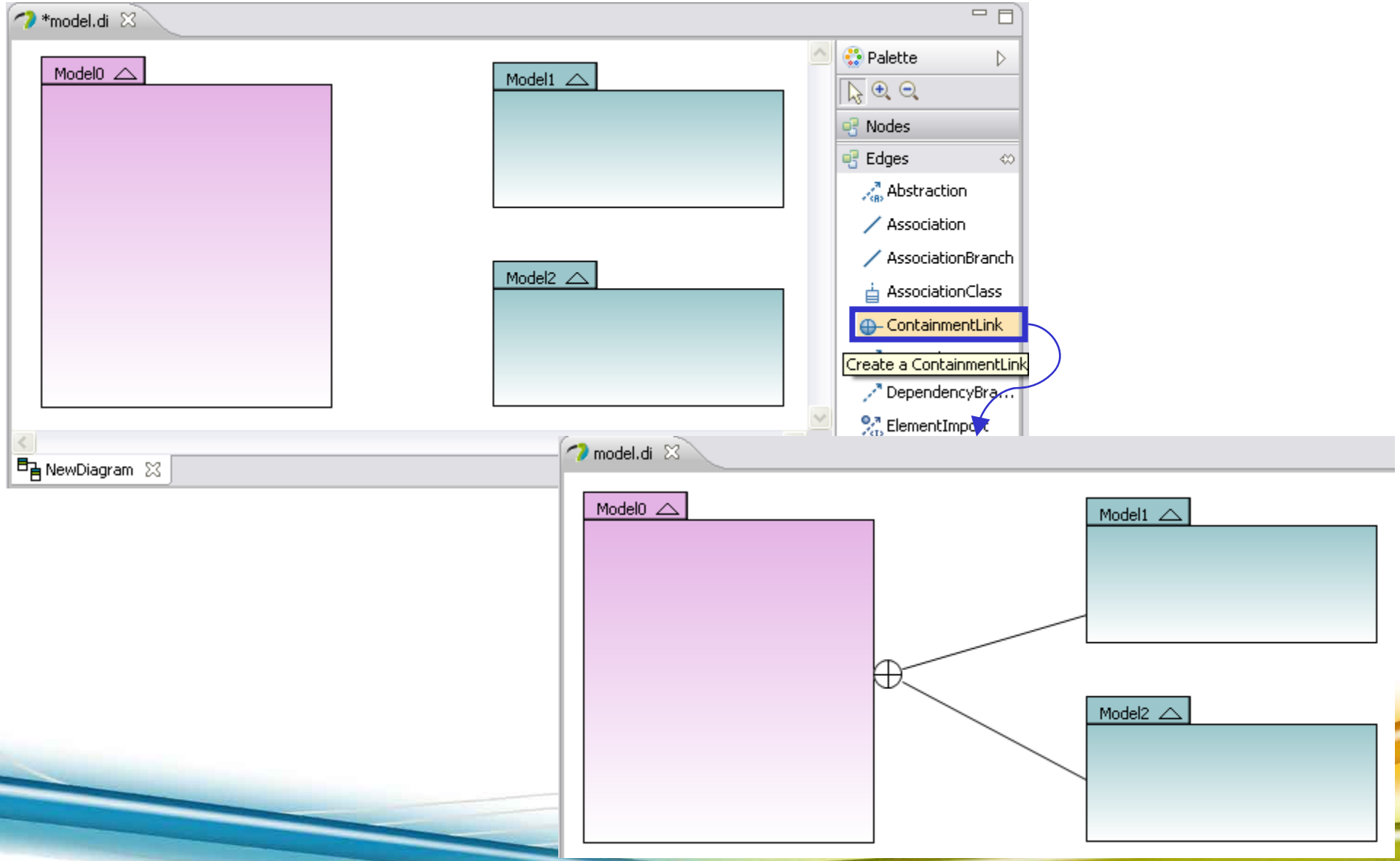
Modeling ClassAssociations

- **Scenario:**
  - Within the palette, use the tool “ClassAssociation”.



Modeling containment relationship

- **Scenario:**
  - Within the palette, use the tool "ContainmentLink"

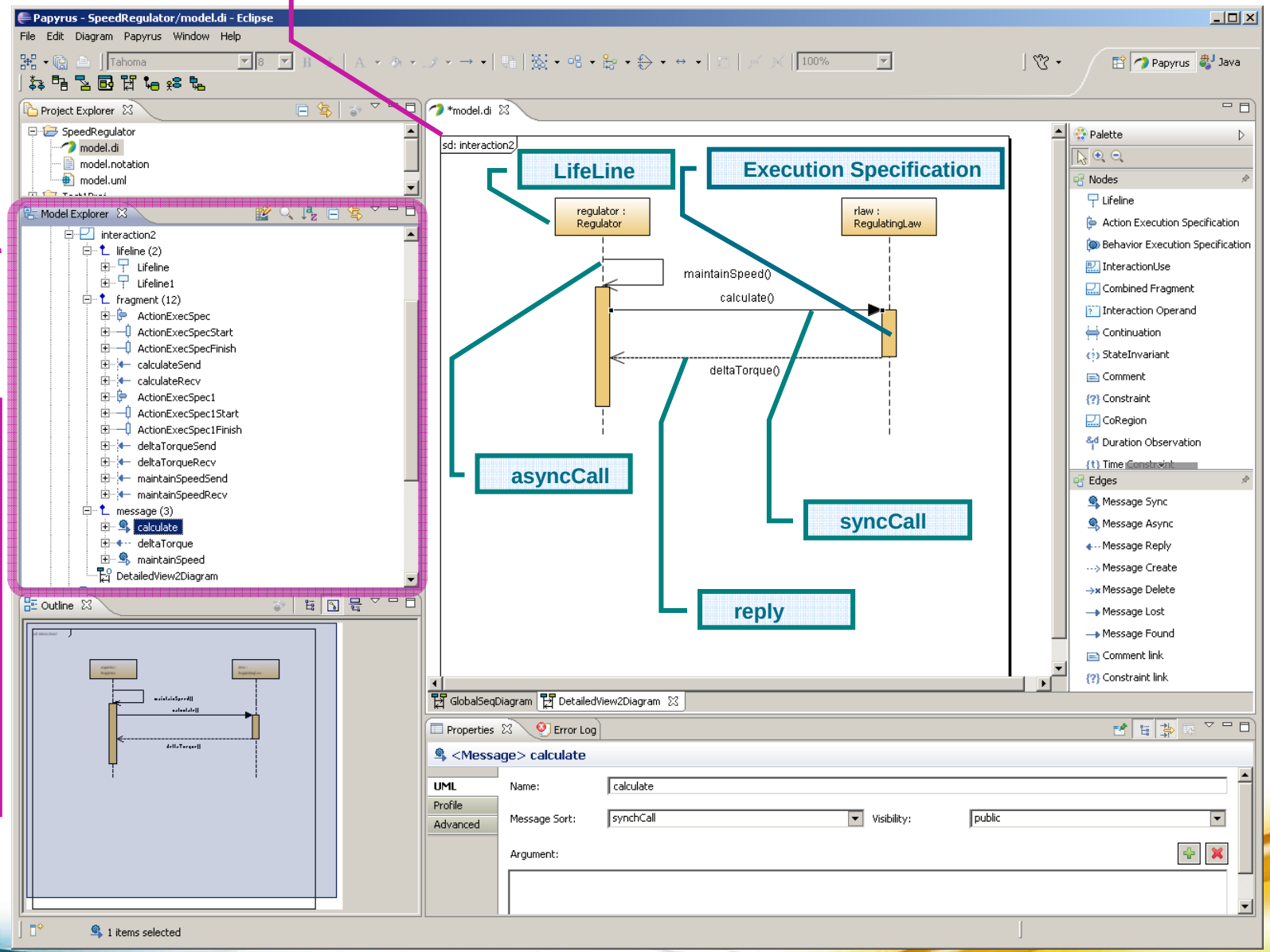


On Sequence Diagrams with papyrus MDT

- **Short Overview of Interaction elements**
- **Creating a first basic sequence diagram**
  - Lifelines
  - Execution Specification
  - Messages
- **Structuring Scnarios**
  - 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



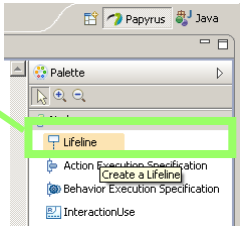
Model elements of interactions

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

UML Sequence diagrams : basics - Lifeline creation

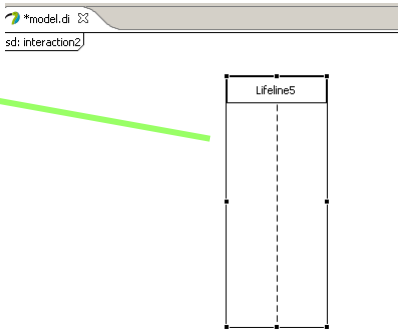
1

Select Lifeline tool in the palette



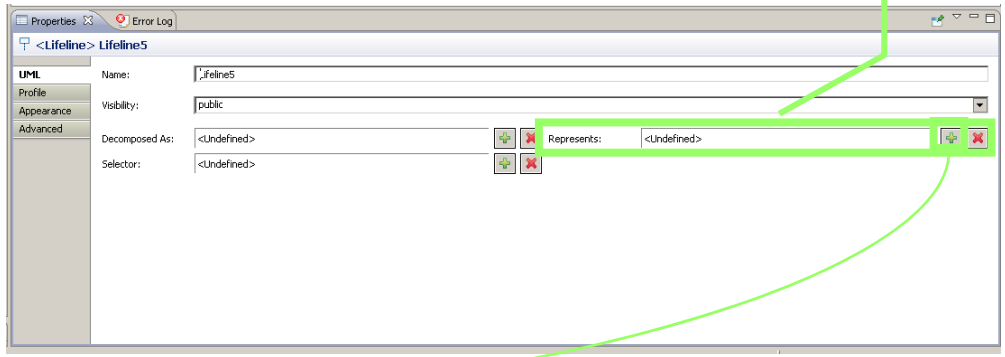
2

Click in the diagram to drop the lifeline



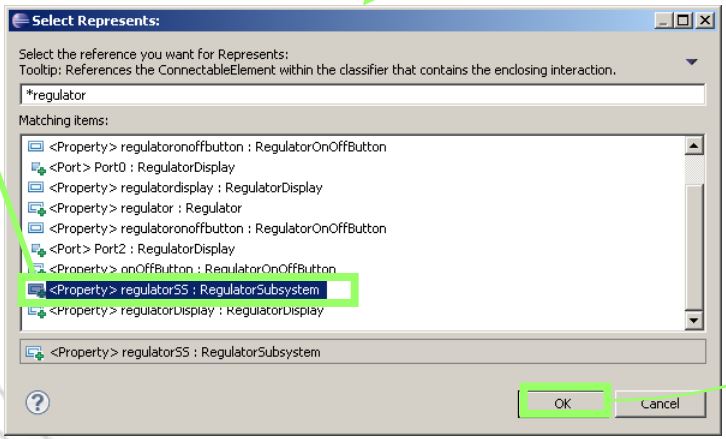
3

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



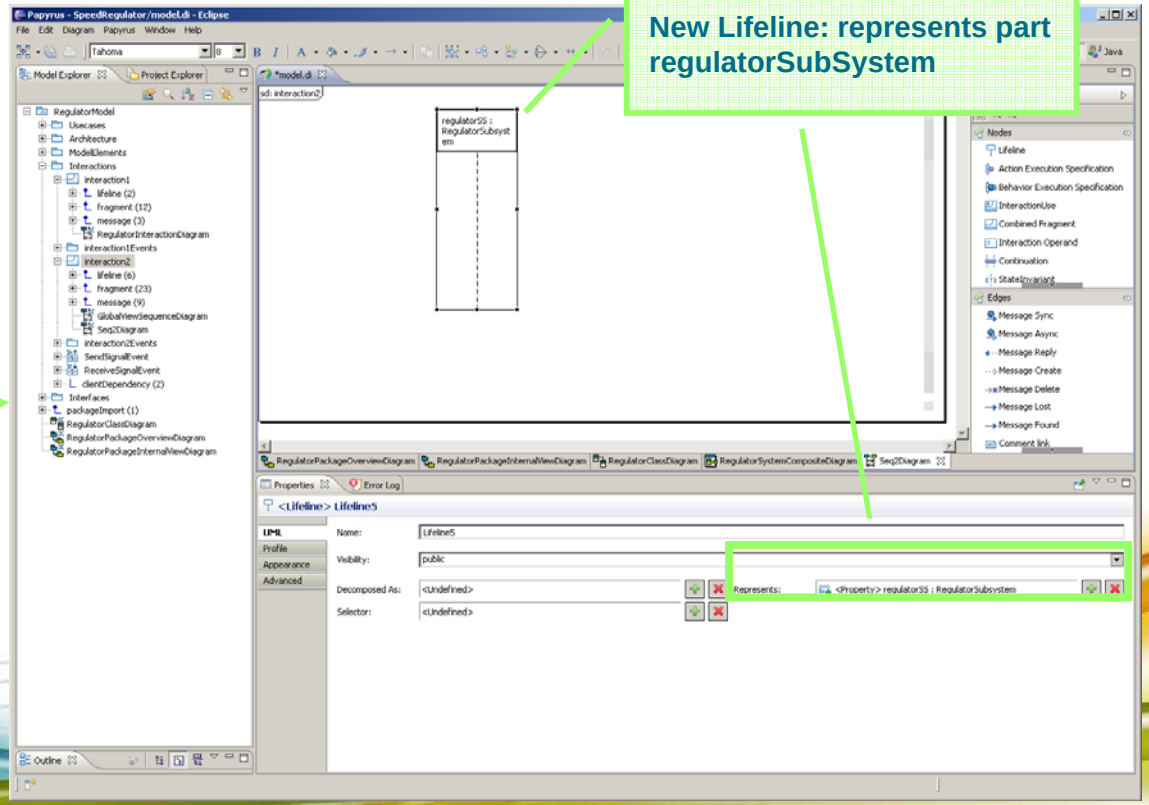
3

Select a part in the pop-up and press OK



4

New Lifeline: represents part regulatorSubSystem

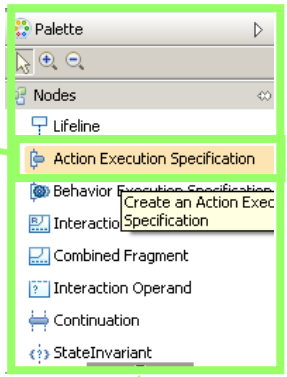


UML Sequence diagrams : basics - Execution Specification creation

- **Two kinds of Execution Specification can be created**
  - Action ES
  - Behavior ES

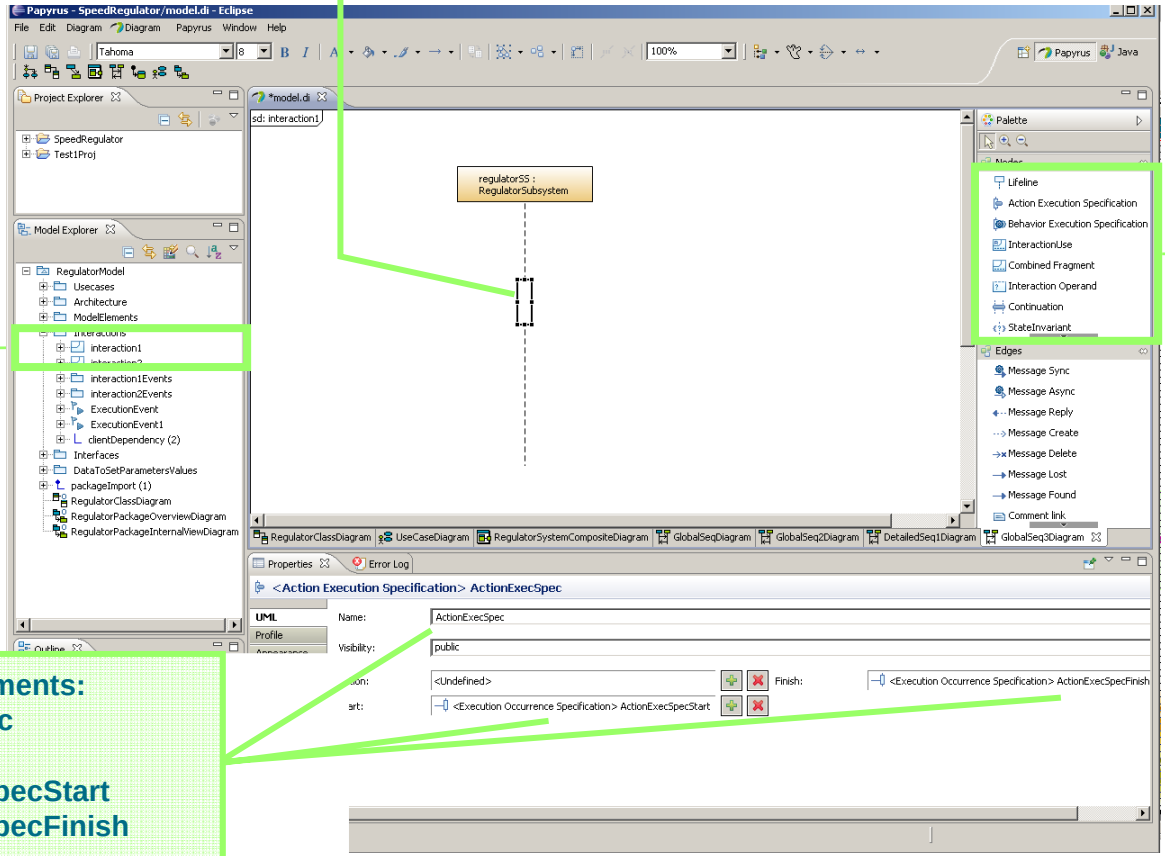
Select ES tool in the palette

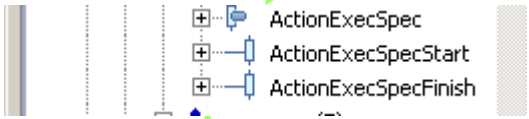
1



2

Click and drop the ES on the lifeline.





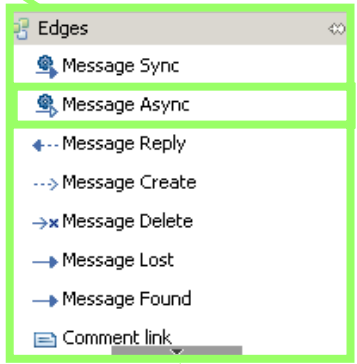
New created UML Elements:
- **ActionExecutionSpec**
- **2 events**
  - **ActionExecutionSpecStart**
  - **ActionExecutionSpecFinish**

UML Sequence diagrams : basics - Message creation(1)

- **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

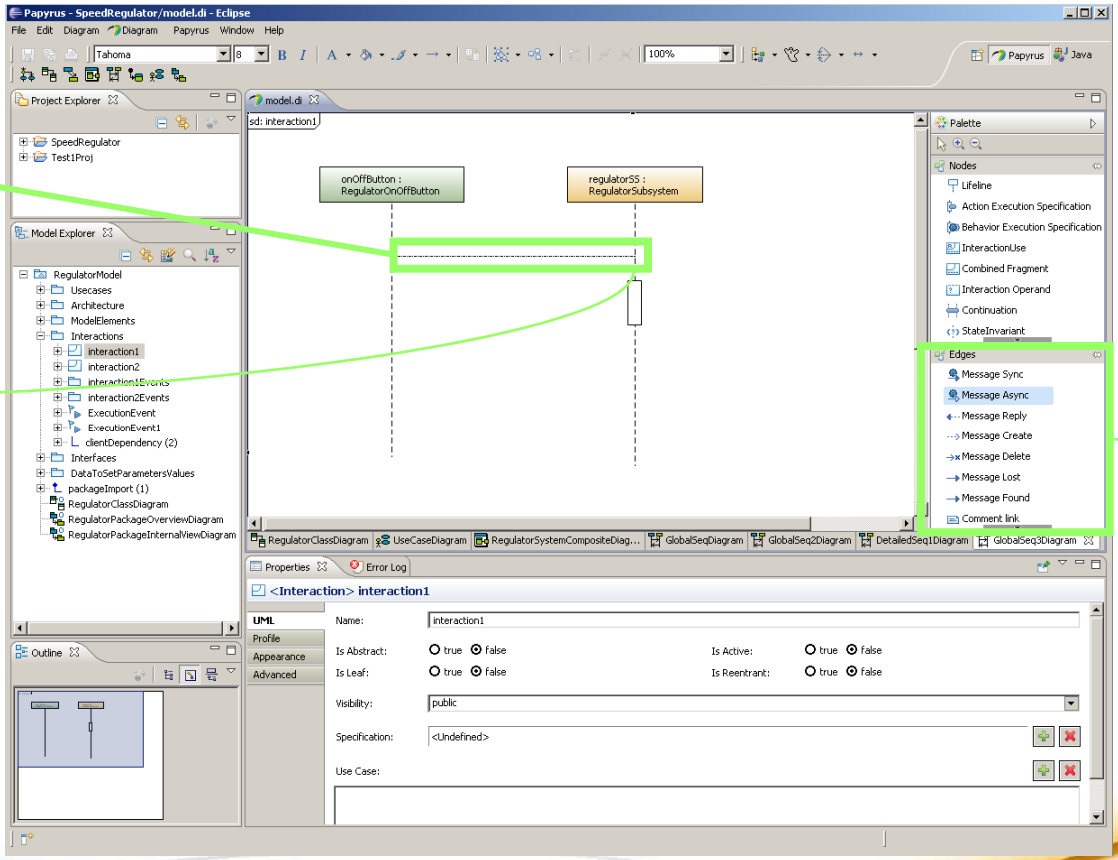
1

Select Message Creation tool from palette.



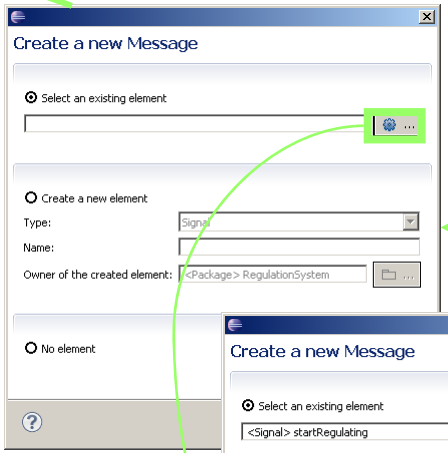
2

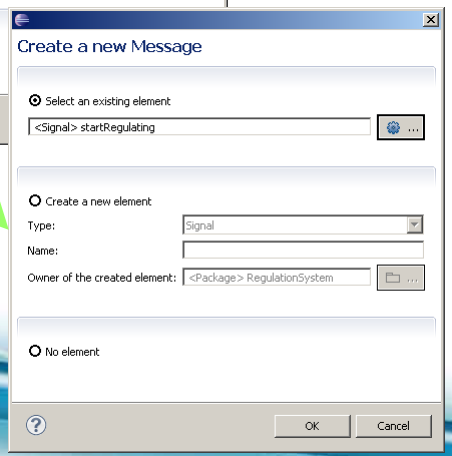
Place the message on the diagram click to set source, maintain mouse button down an drag towards target then release



3

Select an operation or signal or create a new one





UML Sequence diagrams : basics - Message creation(2)

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 propertys 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

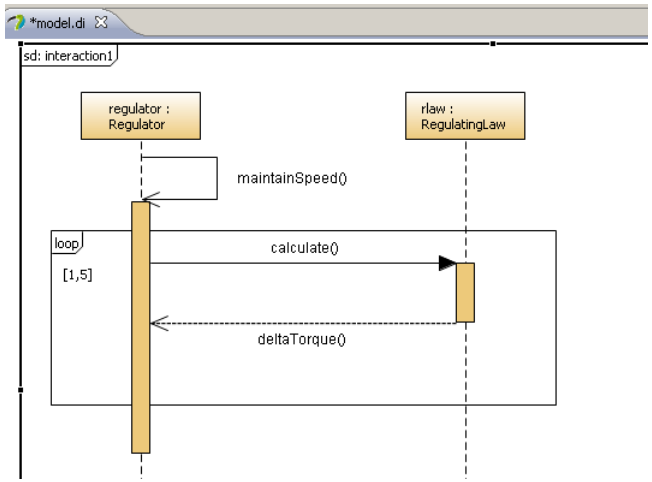
- A reply message can be created only from an ES created by a Sync Message

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

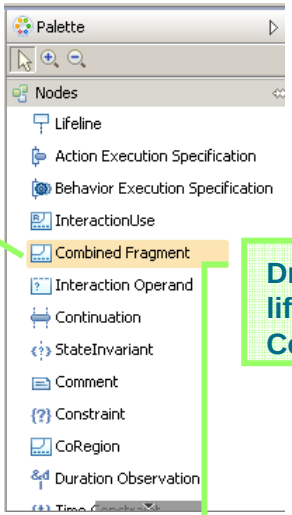
  Steps 4., 5., 6. vary according to the interaction operator selected and specific rules may apply.

UML Sequence diagrams : Combined Fragments Creation(1)

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

1

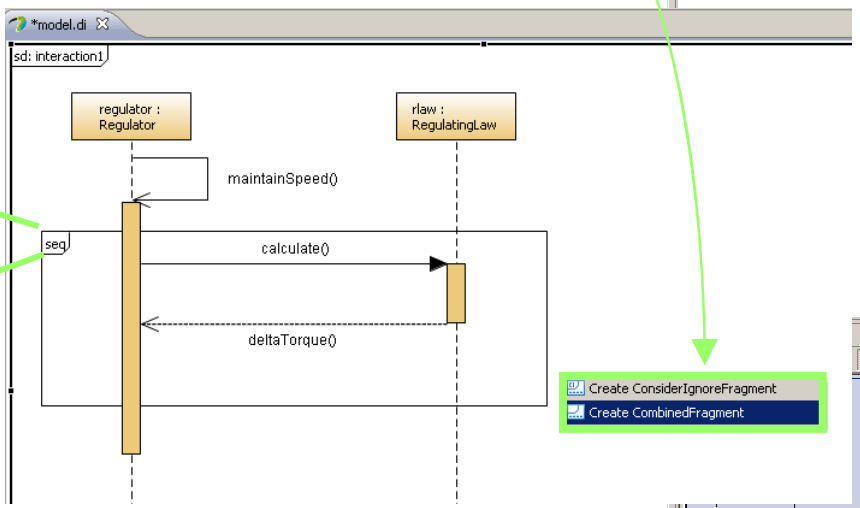
Select CF tool in the palette



2,3

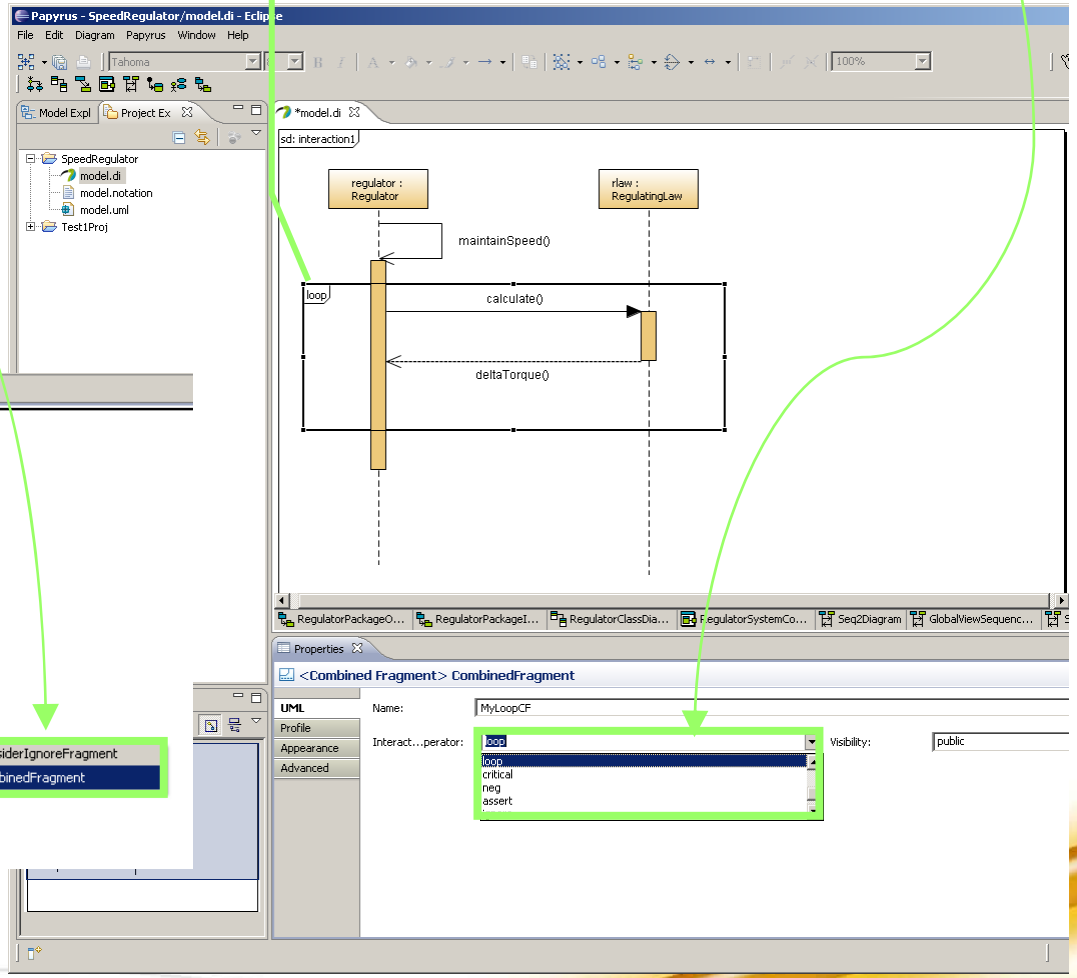
Drop the CF and draw the rectangle on the lifelines. Then select kind of CF Consider/Ignore or Standard CF

New CF with default operand kind = Seq



4

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

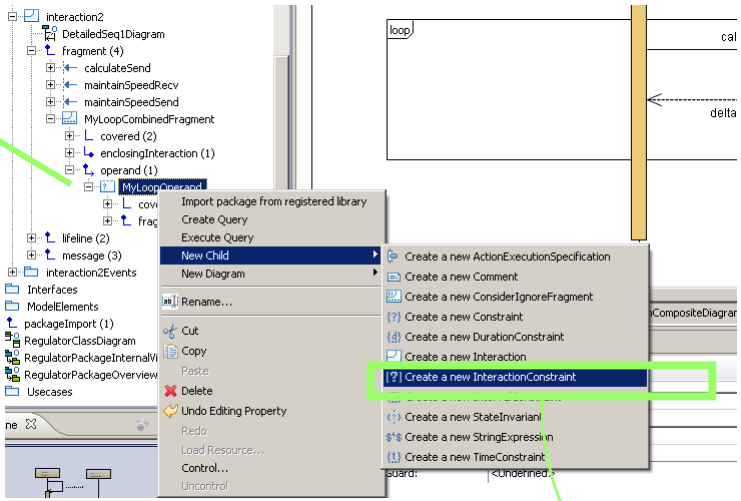


UML Sequence diagrams : Combined Fragments Creation (2)

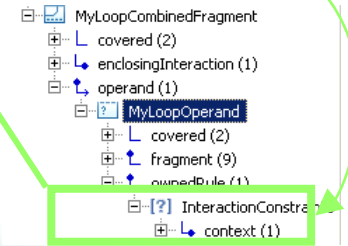
- **Creation steps 5, 6 Setting Guard of the operand**
  - Create Interaction constraint
  - Set Guard constraint in the properties view

5

In the model explorer Create an interaction constraint

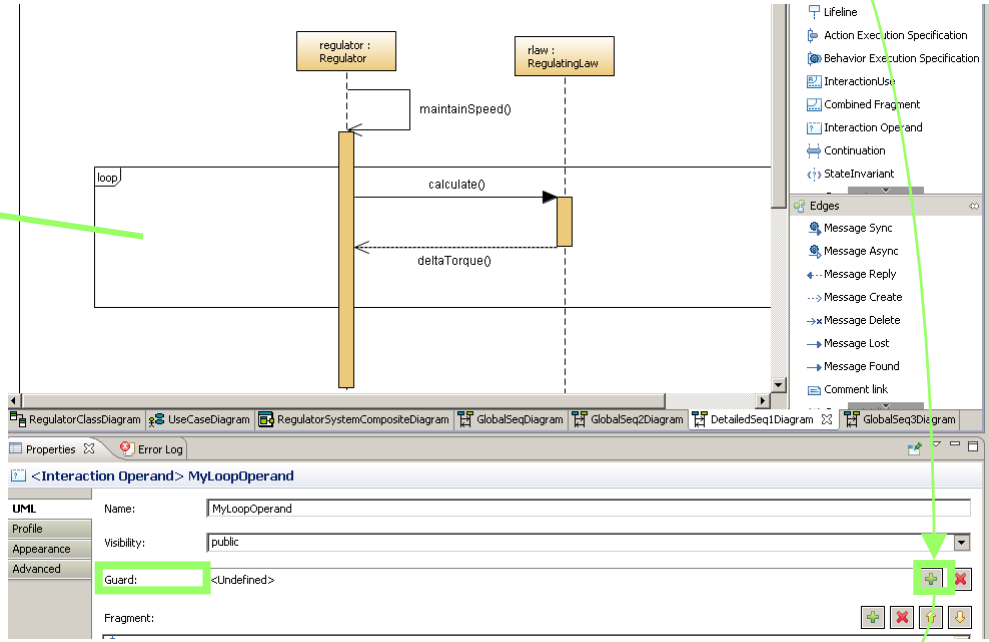


New CF with default operand kind = Seq

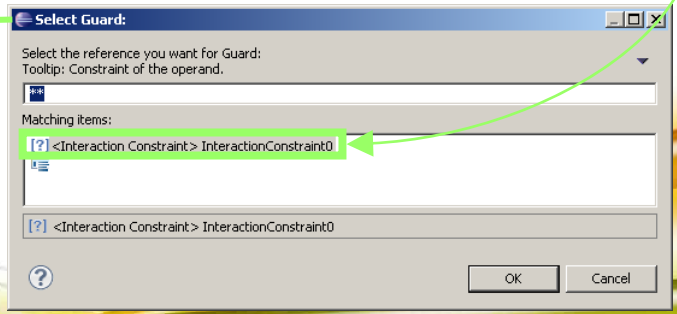


6

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



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

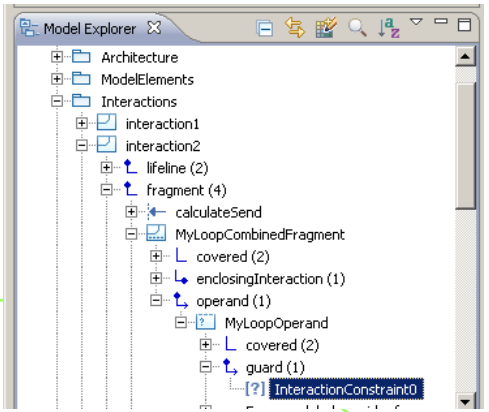
    Procedure to follow
    - 1. create values separately in the model explorer
    - 2. reference these values in the properties view

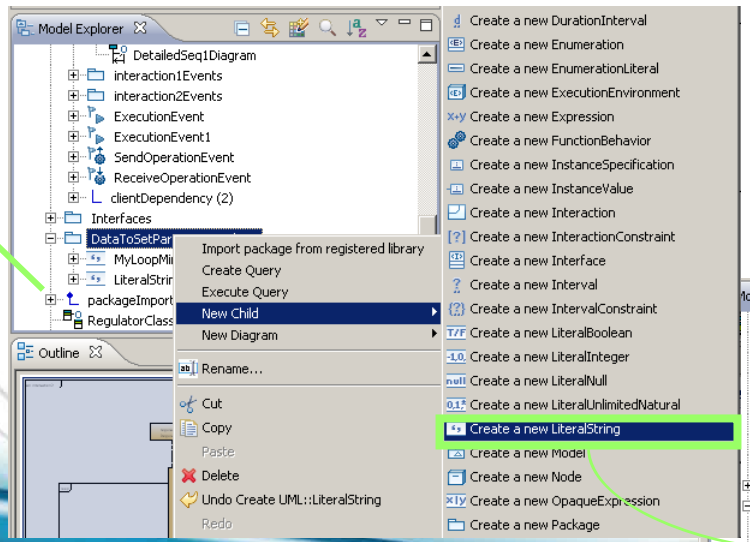
7

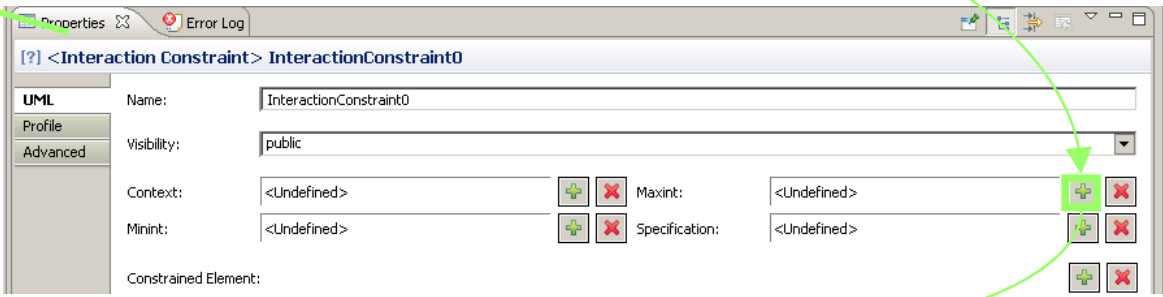
In the model explorer
Create a package to create values that will be referenced by the guard:
- Min
- Max

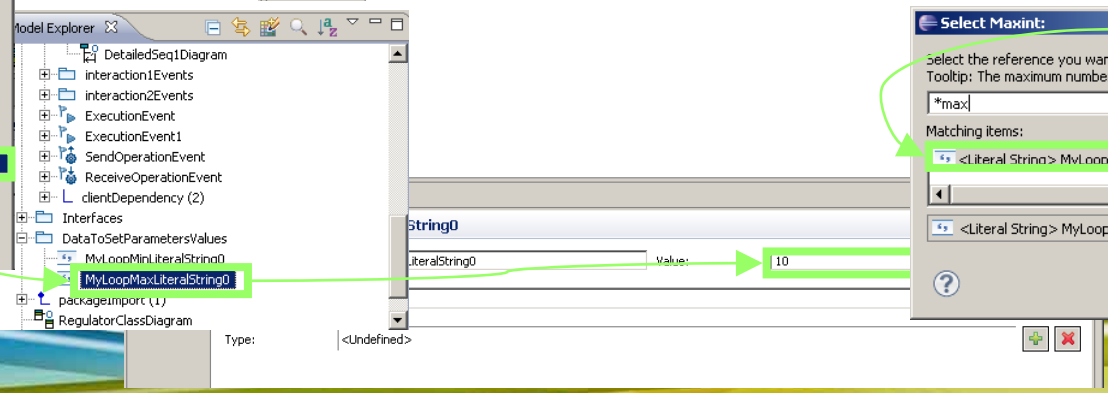
8

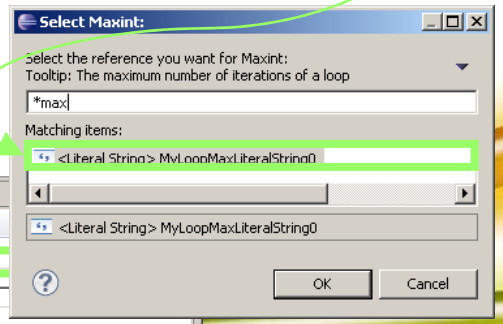
Set Guard properties of Loop Operand
- in model explorer select guard constraint
- in properties view select property to set
(click on green "+" sign on the right)





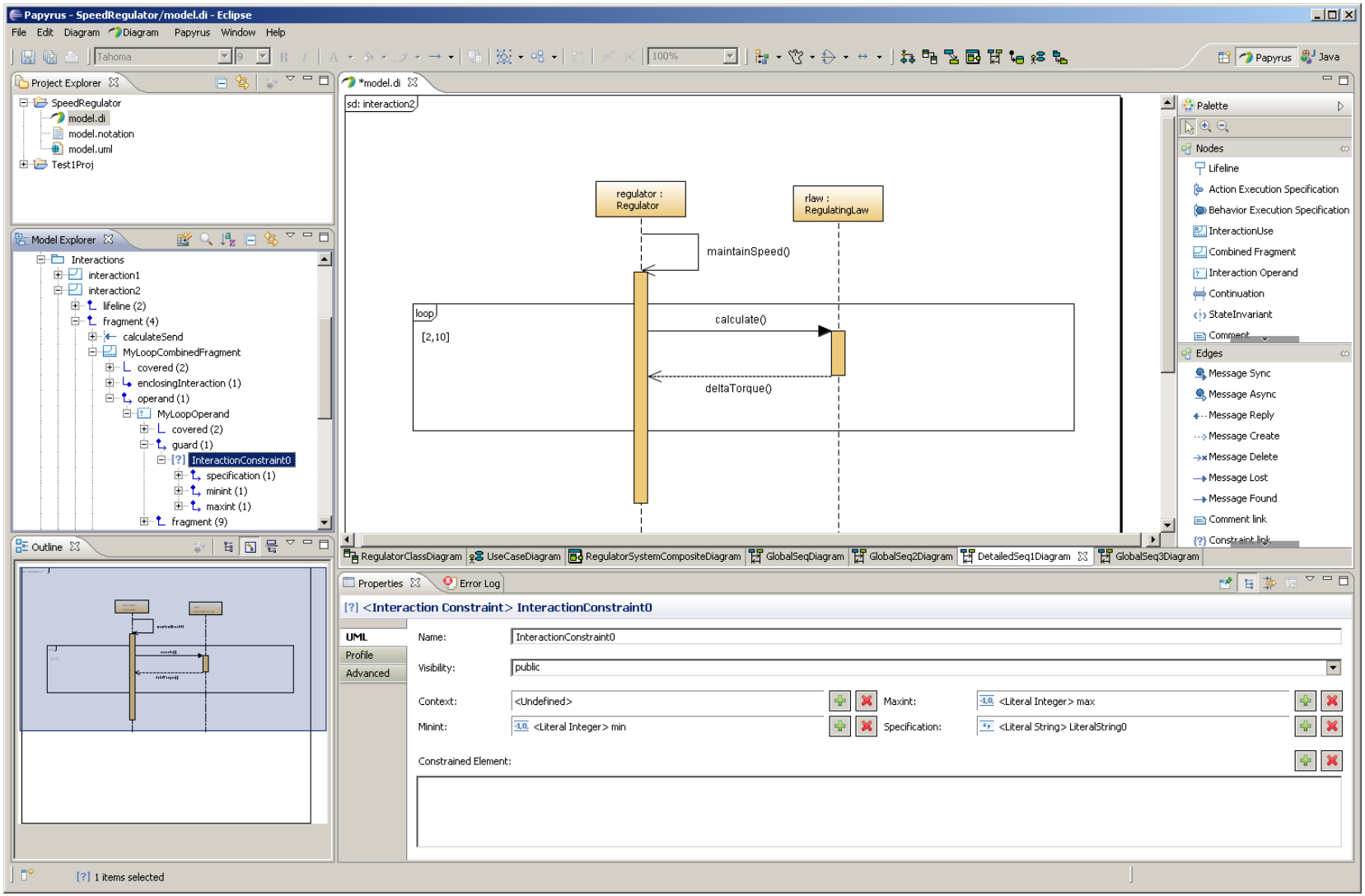






UML Sequence diagrams : Combined Fragments Creation (4)

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

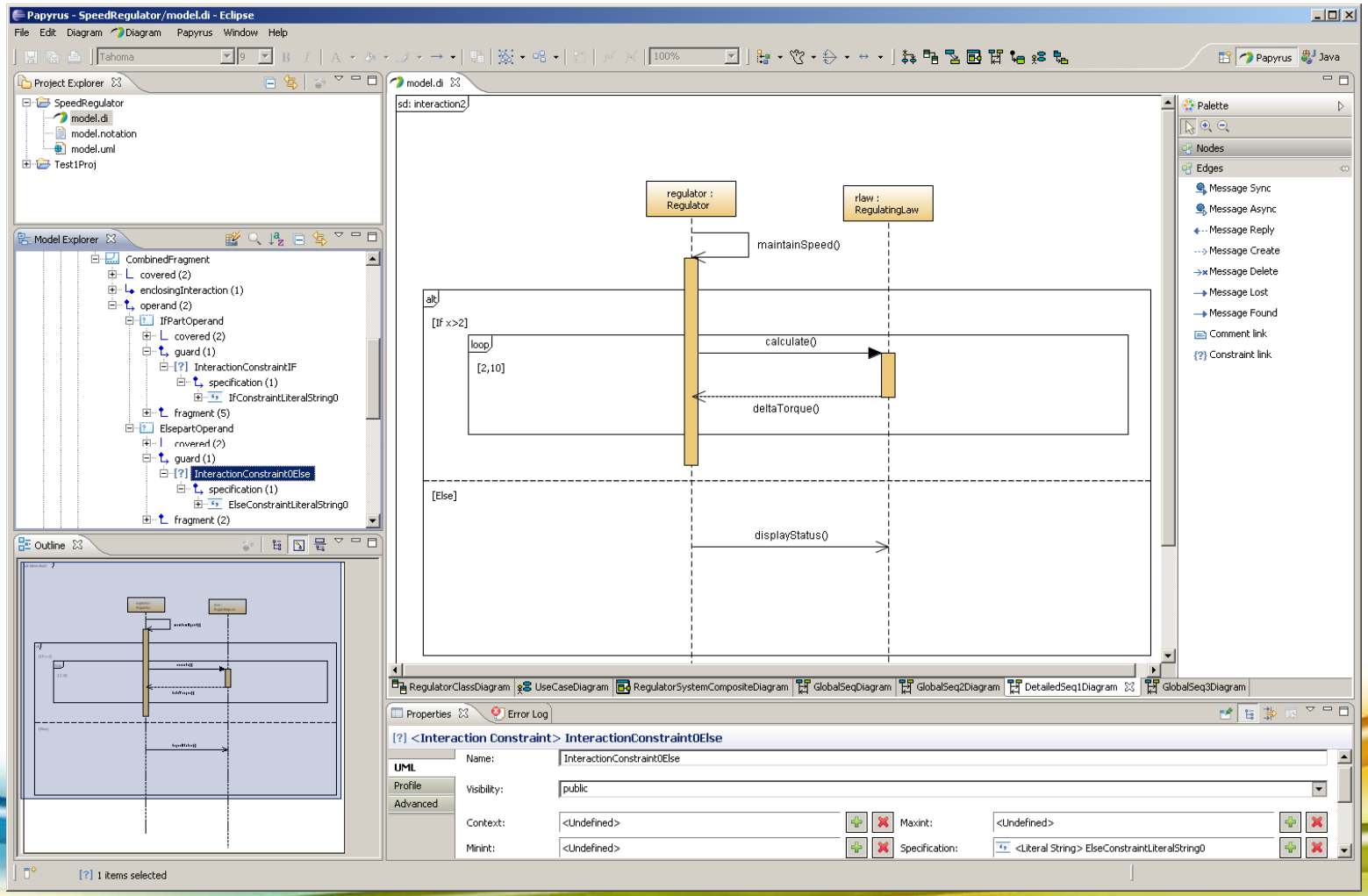


Remark :

- 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)

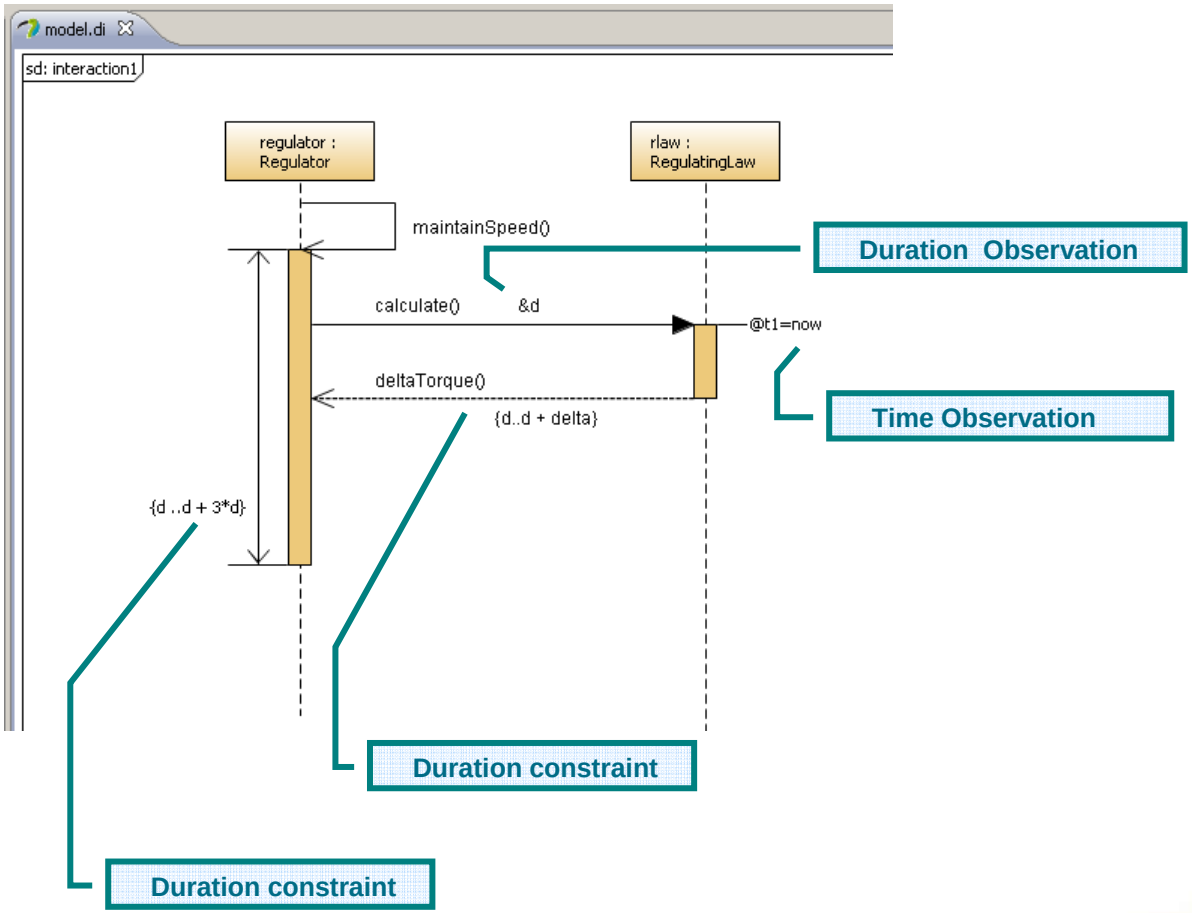
- **Creating Combined Fragments with more than one operand (the alt example)**
- **Same process**
  - Just select a new operand tool in the palette, then click in the operand area
  - A new area appears with a separation line
  - You can enter guards for each operand in the same manner as above (use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

1 Select Duration Constraint tool in the palette

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

3 Set DurationInterval values:
In the model explorer

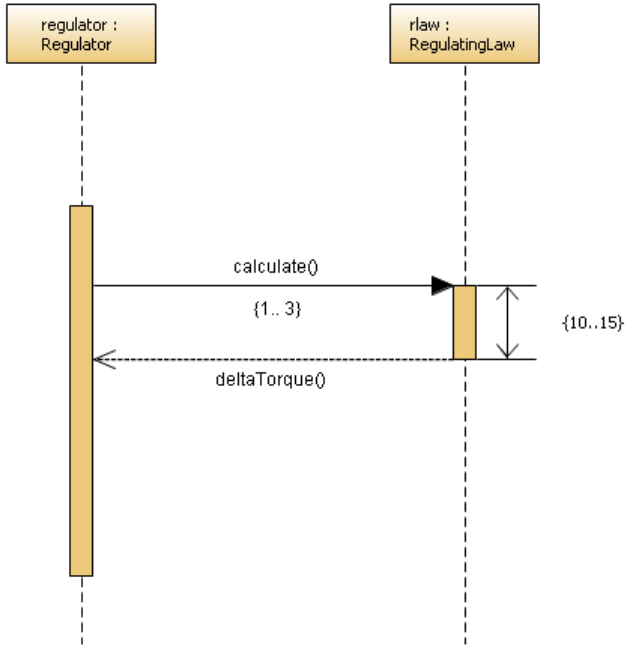
Select Duration Interval
- Min
- Max

4 In properties view

➔ Set value

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 MARTE

Domain specific modeling with UML

- **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.

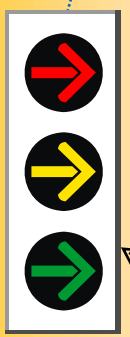
- **Profiles can be used for two different purposes:**
  - To define a domain-specific modeling language.
  - To define a domain-specific viewpoint.

- **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)

Specific notation



« metaclass »
UML::Class

Extenssion

« stereotype »
Semaphore

Specific properties

limit: Integer
getSema: Operation
relSema: Operation

Usage constraint

« Constraint »
limit < UpperLimit

Profile application (User model level)

« semaphore »
SpeedDataLock

SpeedDataLock

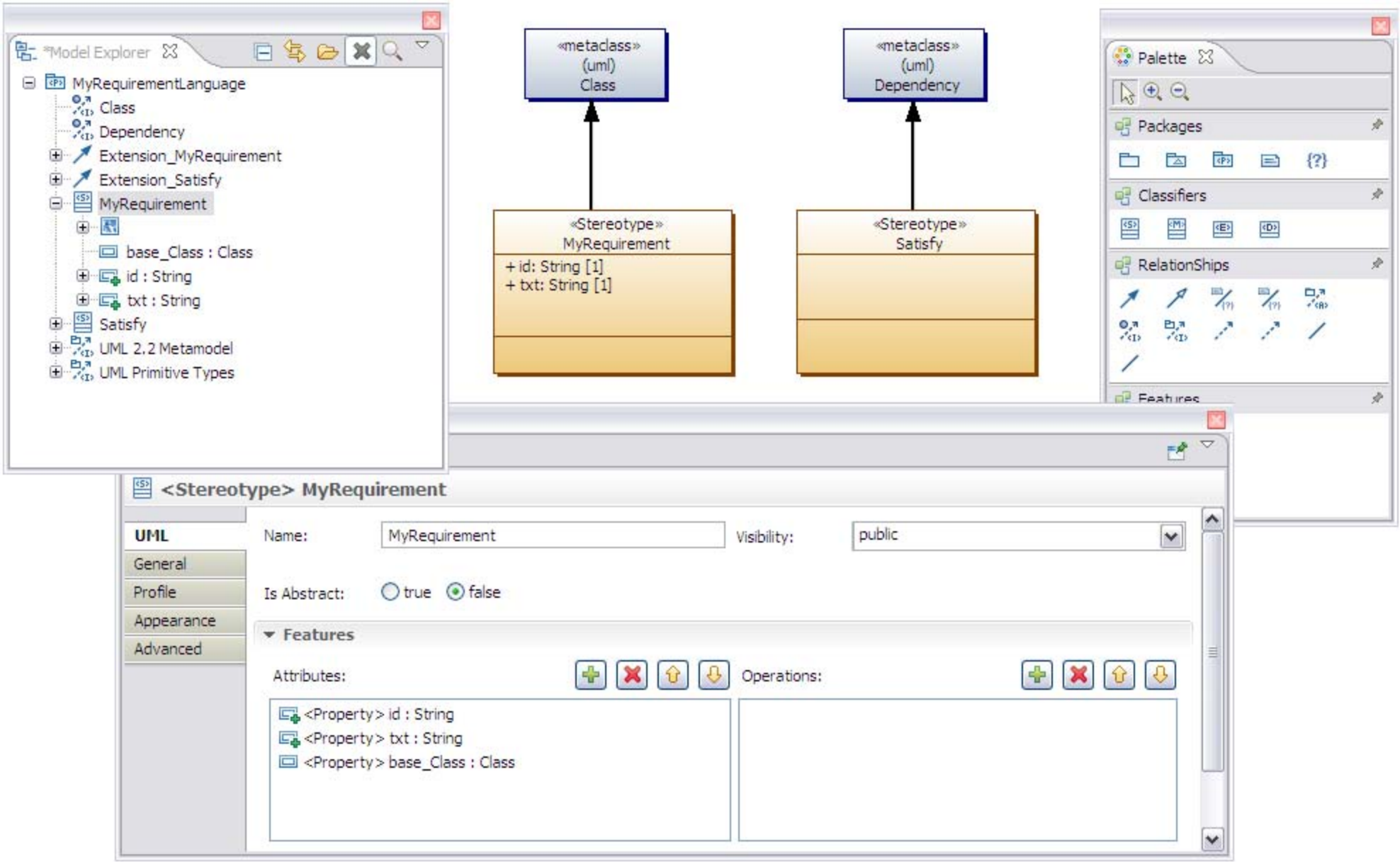
SpeedDataLock

Customizations for DSL

- **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

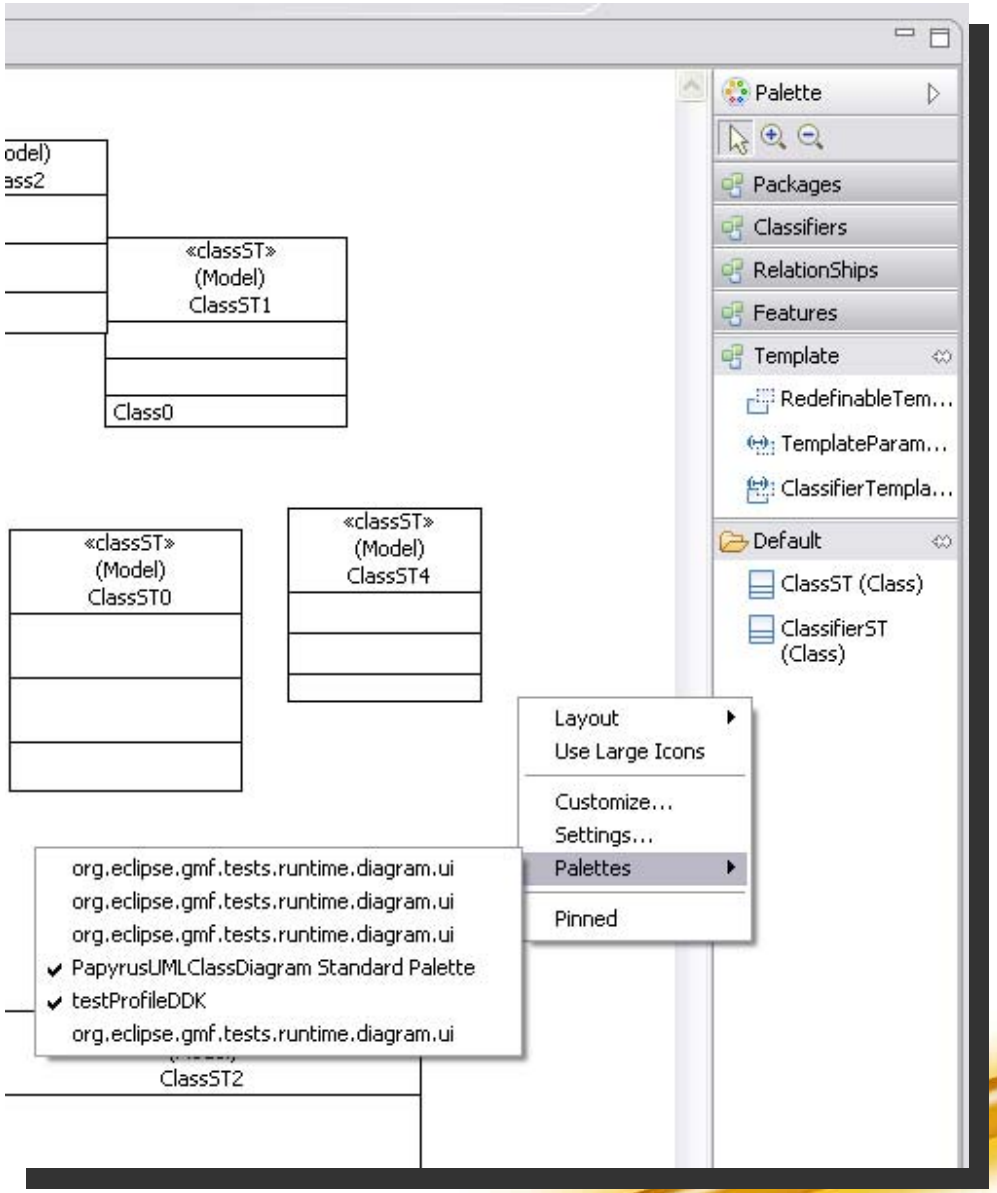
Customizations for DSL

- **Language support with profile**



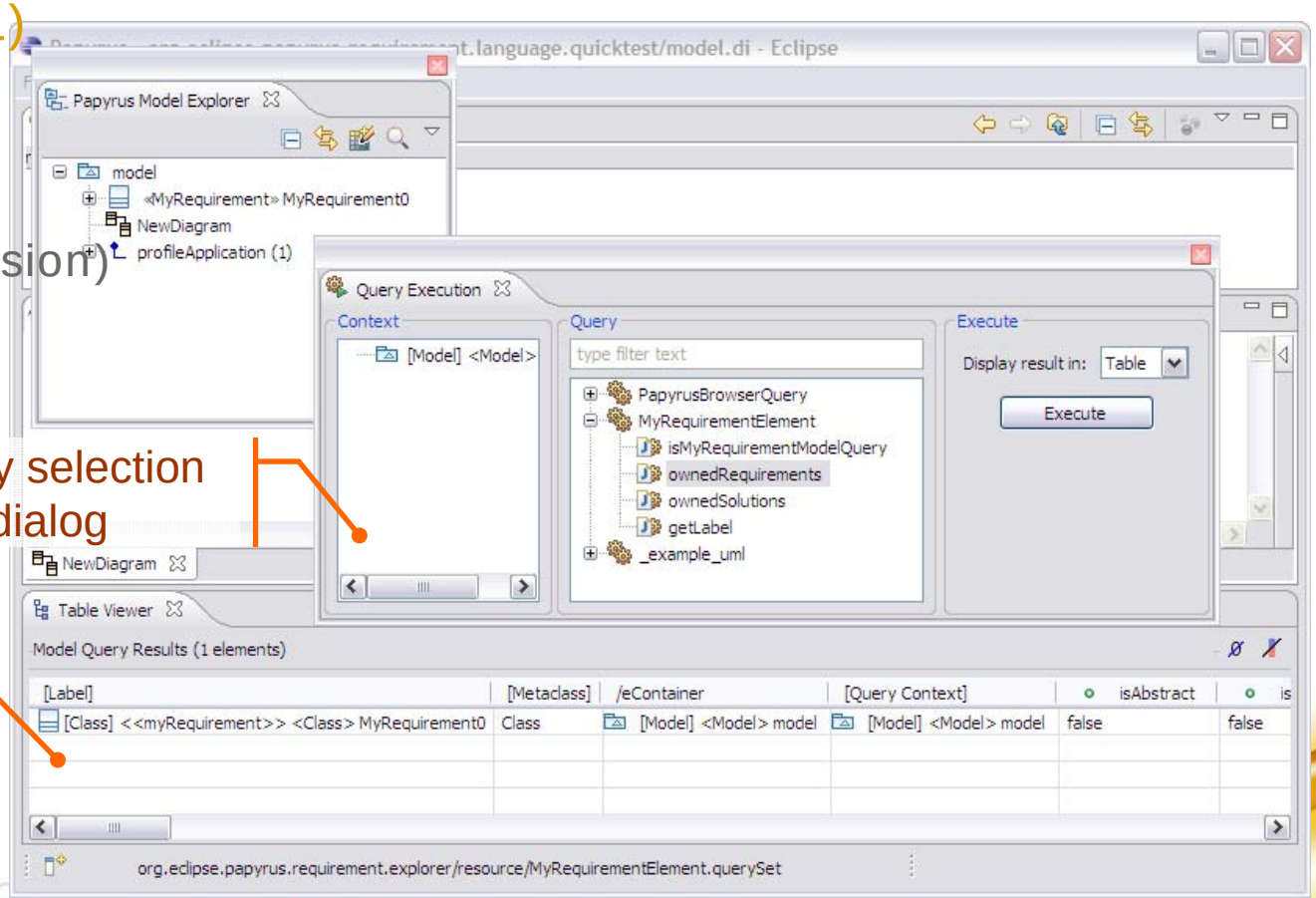
Customizations for DSL

- **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



Customizations for DSL

- **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)

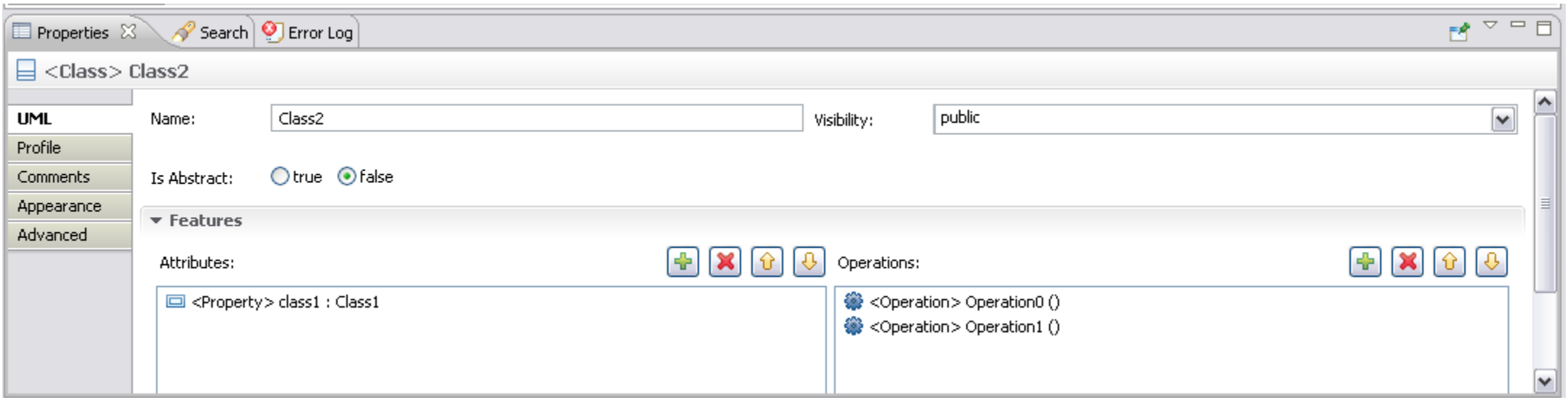


Query selection dialog

Query result visualization

Customizations for DSL

- **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

Customizations for DSL

- **Providing dedicated property views (2/3)**

Customize property view Wizard

Content:

[EClass] BehavioralFeature -> Namespace, Feature [org.eclipse
[EClass] BehavioredClassifier -> Classifier [org.eclipse.uml2.uml.
[EDataType] Boolean [boolean]
[EClass] BroadcastSignalAction -> InvocationAction [org.eclipse.
[EClass] CallAction -> InvocationAction [org.eclipse.uml2.uml.Ca
[EClass] CallBehaviorAction -> CallAction [org.eclipse.uml2.uml.(
[EEnum] CallConcurrencyKind
[EClass] CallEvent -> MessageEvent [org.eclipse.uml2.uml.CallE
[EClass] CallOperationAction -> CallAction [org.eclipse.uml2.uml
[EClass] CentralBufferNode -> ObjectNode [org.eclipse.uml2.um
[EClass] ChangeEvent -> Event [org.eclipse.uml2.uml.ChangeEv
[EClass] Class -> EncapsulatedClassifier, BehavioredClassifier [o
Section Set for single NameElement: org.eclipse.uml2.uml.Na
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

Customizations for DSL

- **Providing dedicated property views (3/3)**

Papyrus - Debug/model.di - Eclipse Platform

Properties | Search | Error Log

<Class> Class2

UML | Profile | Comments | Appearance | Advanced

Name: Class2 Visibility: public

Is Abstract: true false

Features

Attributes: <Property> class1 : Class1

Operations: <Operation> Operation0 () <Operation> Operation1 ()

Properties | Search | Error Log

<<eClass>> <Class> Class0

UML | Profile | Comments | Appearance | Advanced

Name: Class0 Visibility: protected

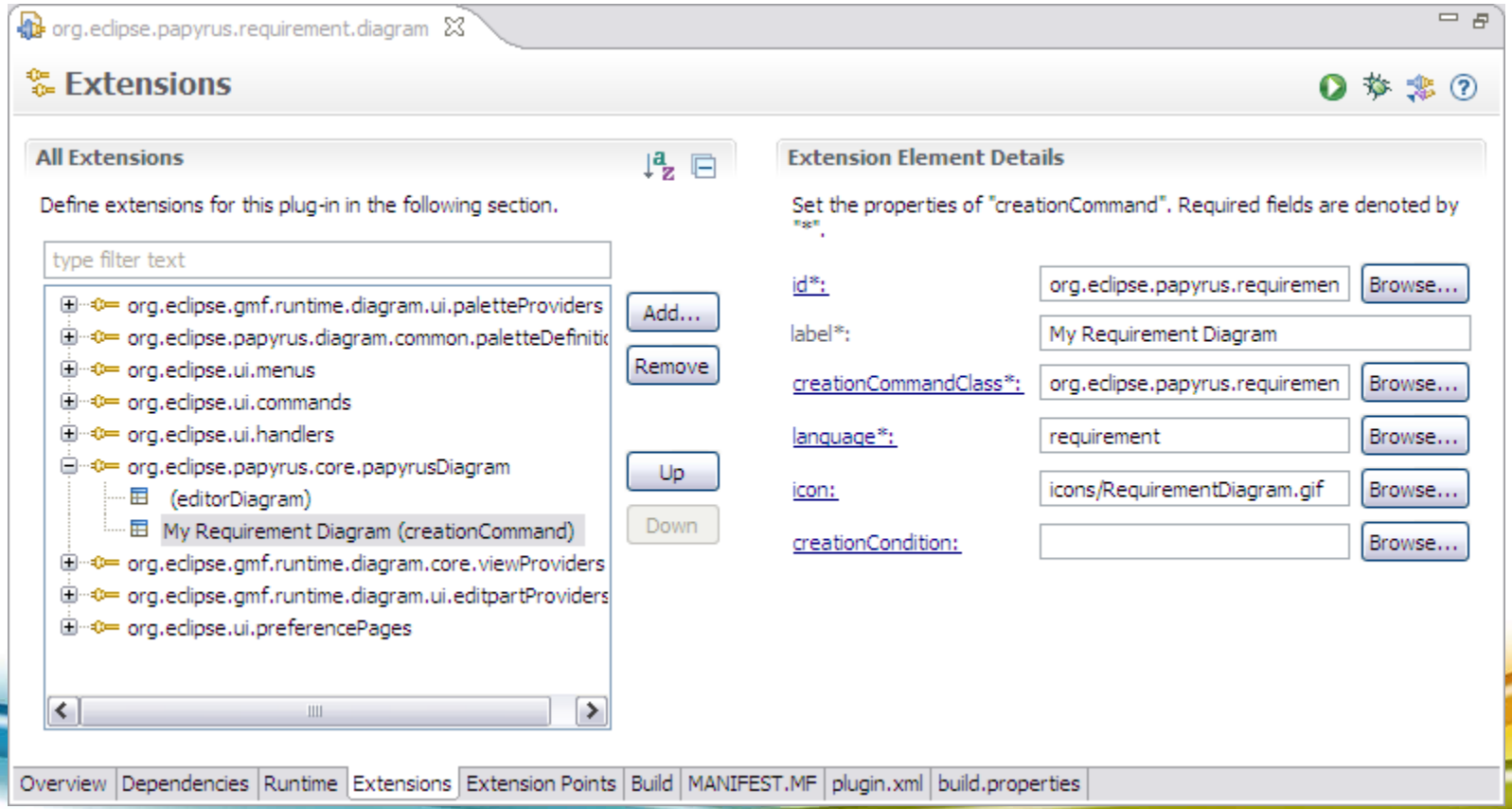
EClass

Class Name: Serialization

XML Co... kind: <Unset> Unspecified Empty Simple Mixed ElementOnly

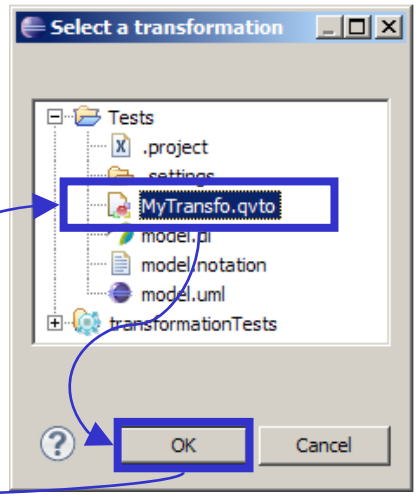
Customizations for DSL

- **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

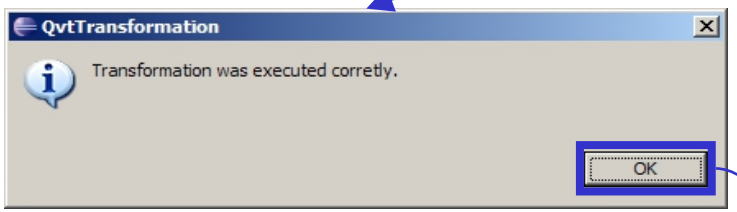


Execution of QVTo transformations in Papyrus (seq.)

- **Launching the transformation**
  - Scenario:
    - Within the editor, right-click and select following action:

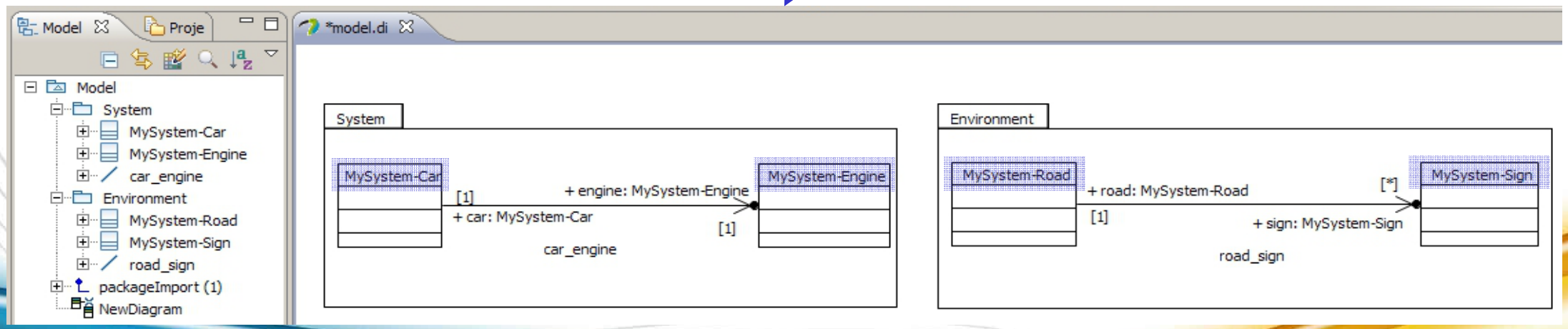


- **Message to confirm (or an error dialog with diagnostic if it failed):**



Classes have been renamed according to the model transformation definition.

- **Resulting model:**



Summary on model transformation within Papyrus

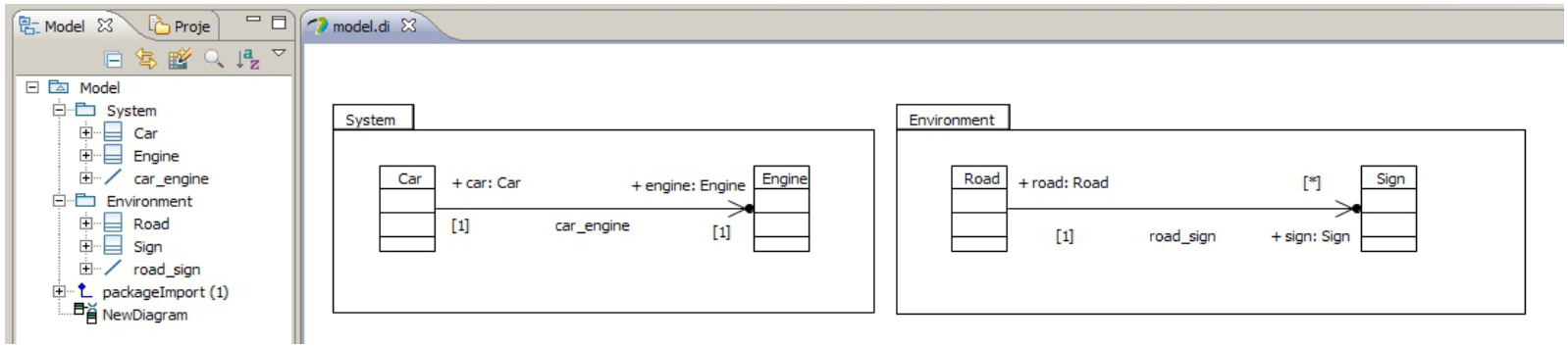
- **Benefits**
  - The transformation is executed using Papyrus' editingDomain.
  - Therefore, the transformation is a considered as a regular command:

> Modifications performed though the model transformation can be undone/redone!

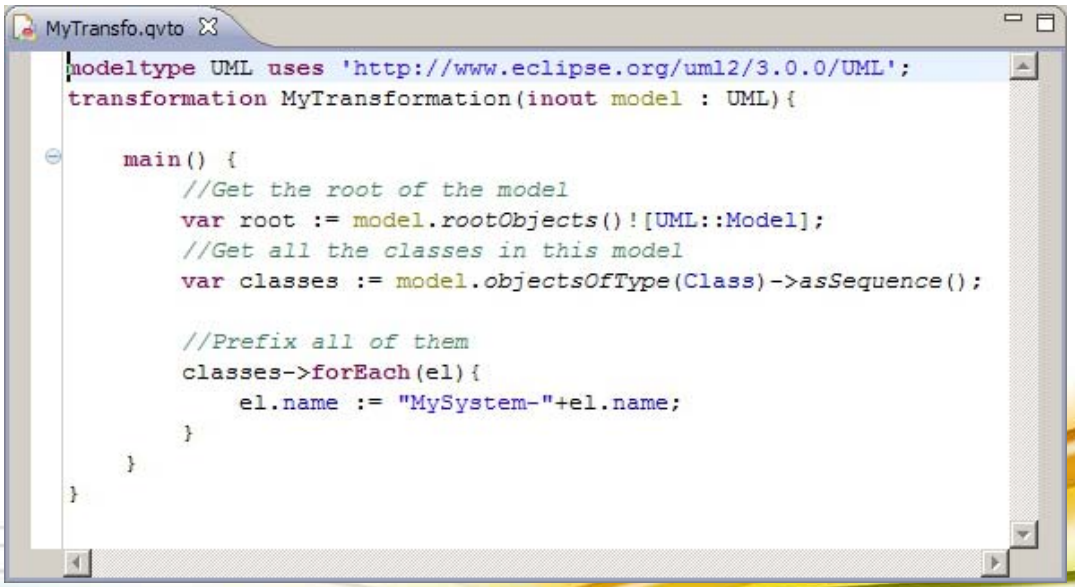
- **Current limits**
  - Transformations signature:
    - Transformations must have only one INOUT parameter,
    - And the metamodel for this parameter must be UML.
  - If the transformation has OUT and other IN parameters, run directly with QVTo
    - ➔ The transformation cannot be undone and redone!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



Conclusion and perspectives

- **First release -> 0.7.0 (Mid-July)**
- **Next steps: next releaase 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!

More Information

- **For developers...**
  - 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**