

**TOPCASED**

Toolkit in Open-Source for Critical Application & Systems Development

# TOPCASED 2.5 UML Editor tutorial

Raphaël Faudou



# Agenda

- What's new in this tutorial ?
- Create a new TOPCASED project/model
- Edit a diagram
- Document the model elements
- Autoresize the diagram
- Export the diagram as an image
- Add a diagram
- Adjust the structure of the model
- Validate a model
- Specialization of items – UML profile
- Navigation in the diagrams
- Version management of models
- Collaborative management of models
- Work in « out-sourcing » mode
- Search the models



## What's new in this tutorial ?

- Multi creation element ([slide 10](#))
- Customize figure's element using stereotype ([slide 29](#))
- Validate a model ([slide 23](#))
- You can consult the release note to find all news on TOPCASED 2.5:
  - ▶ <http://gforge.enseeiht.fr/frs/download.php/2003/ReleaseNote2.4.0.pdf>

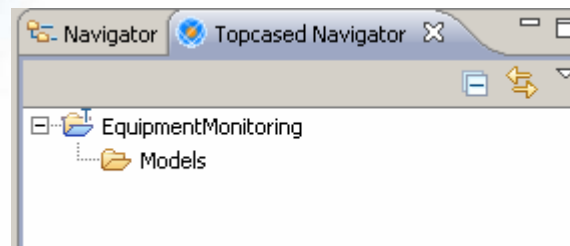


## Create a TOPCASED project

- TOPCASED Perspective
  - ▶ Helps in creating the project



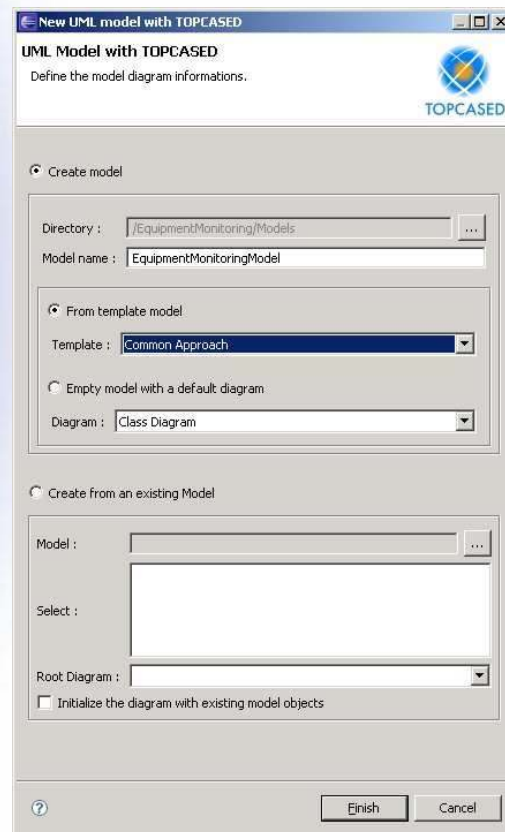
- A project of type « TOPCASED » is created with a « models » directory





## Create a new model (1/2)

- Blank model or a model created based on a template (structure and diagrams)
  - ▶ Different templates according to the editor
  - ▶ Possibility of adding other templates (plug-in)





## Create a new model (2/2)

- ▶ Two files

*Model data (stored in xxx.uml)*

*Graphical model-based editor*

*Diagram data (stored in xxx.uml, with reference to xxx.uml)*

The screenshot displays the Topcased UML editor interface. The **Navigator** view shows a project structure with a **Models** package containing **EquipmentMonitoringModel.uml** and **EquipmentMonitoringModel.uml,di**. The **Outline** view shows a hierarchical structure of UML models and diagrams, including **Use Case Diagram Main**, **Class Diagram Main**, and **Component Diagram Main**. The **Diagram Main** view shows a **package Logical View** diagram. The **Properties** view shows the **Diagram Main** properties, including **Diagram Link**, **Name**, **Position**, **Reference**, **Size**, **Viewport**, **Visible**, and **Zoom**.

Property	Value
Diagram Link	
Name	Main
Position	0,0
Reference	
Size	100,100
Viewport	0,0
Visible	true
Zoom	1.0

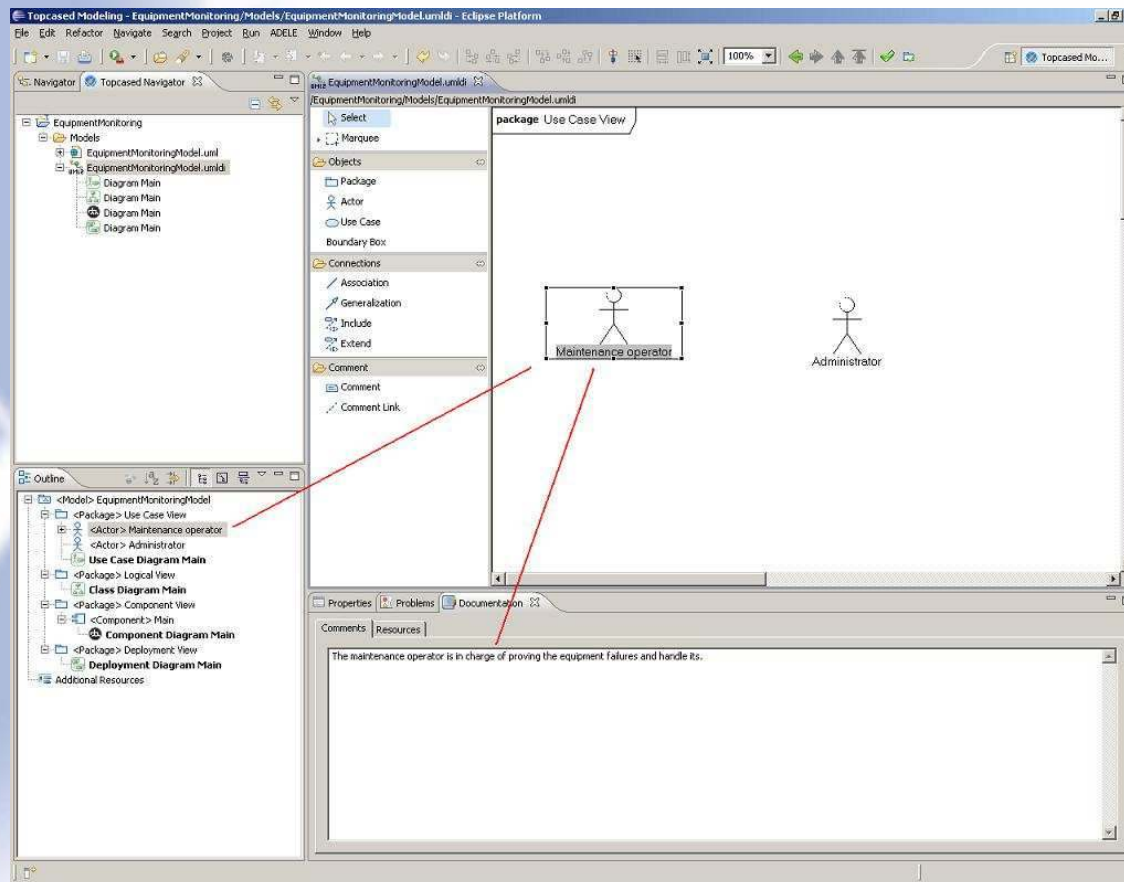
*UML model and diagram data, available in the outline view*

*Property view, contextual to the selection*



## Edit a diagram (1/8)

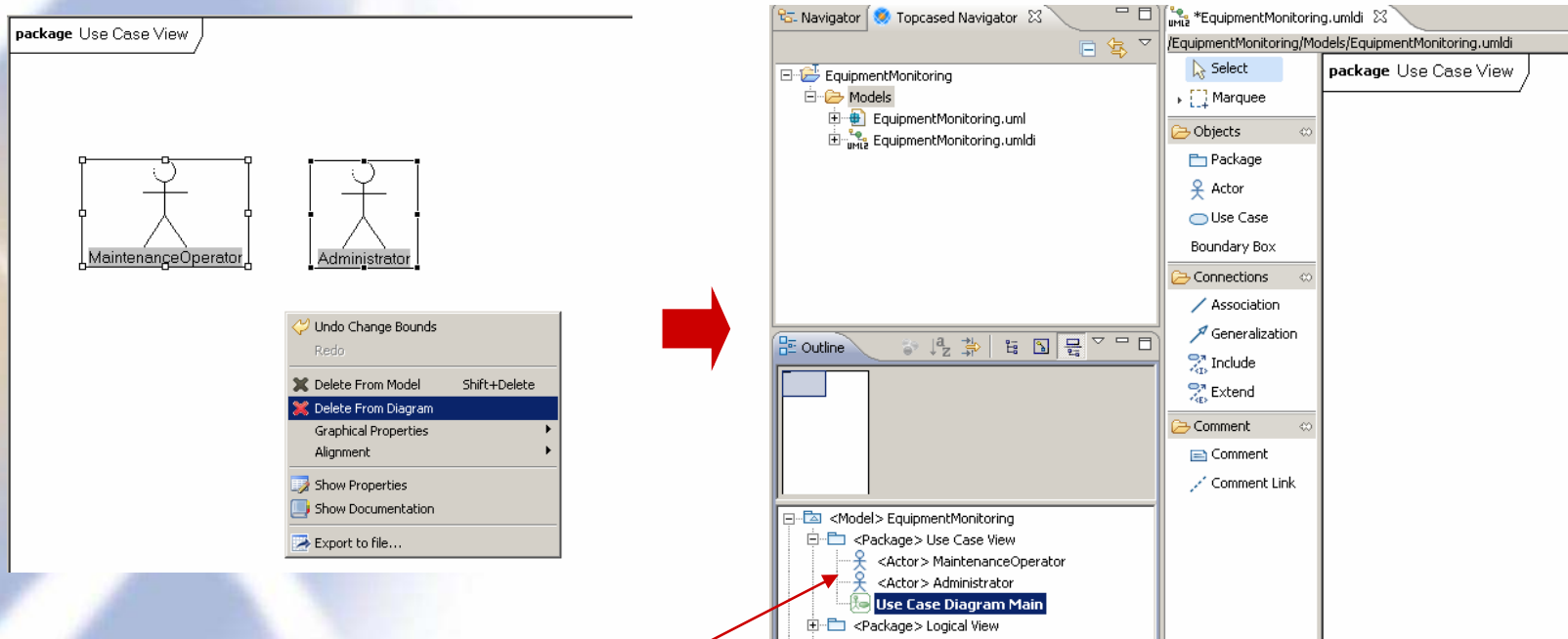
- Open the use case diagram (double click on it in the outline view)
  - ▶ Create an actor via the palette : an item is added to the « outline » view (model)
  - ▶ The documentation view enables to comment on current item





## Edit a diagram (2/8)

- Dealing with mistakes
  - ▶ Let us suppose we want first to focus on use cases
  - ▶ We can delete actors from the current diagram



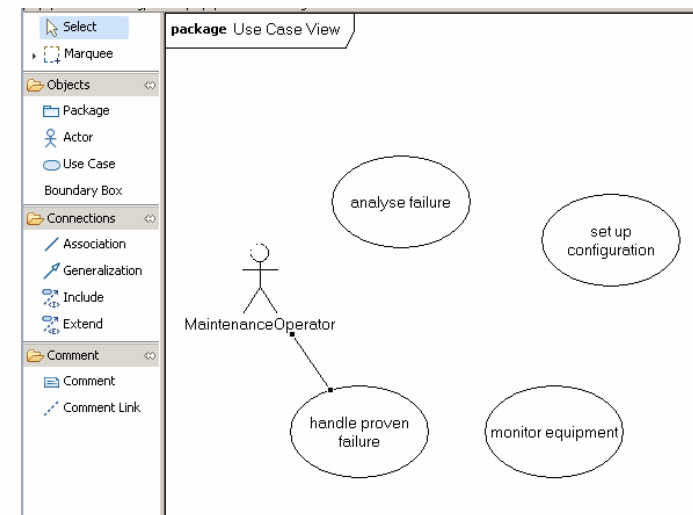
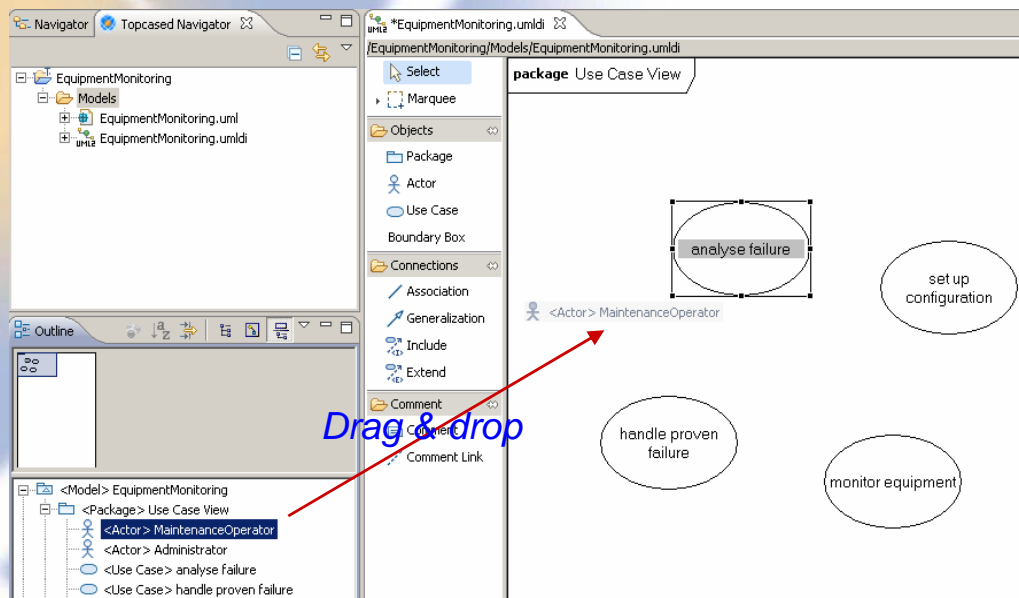
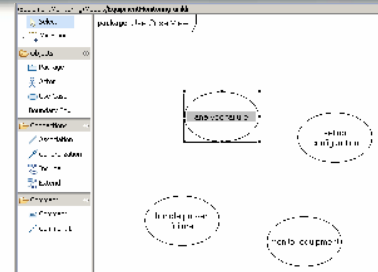
- ▶ Please notice that both actors remain visible in the model (outline view)
  - It is because we deleted them from diagram only (not from the model)





## Edit a diagram (3/8)

- Now let us create use cases
  - ▶ from the palette
- Linking actors and UC
  - ▶ We can create new actors from the palette...
  - ... or retrieve existing ones from the model (outline view) by drag-and-drop
  - ▶ And then create association between actors and use cases

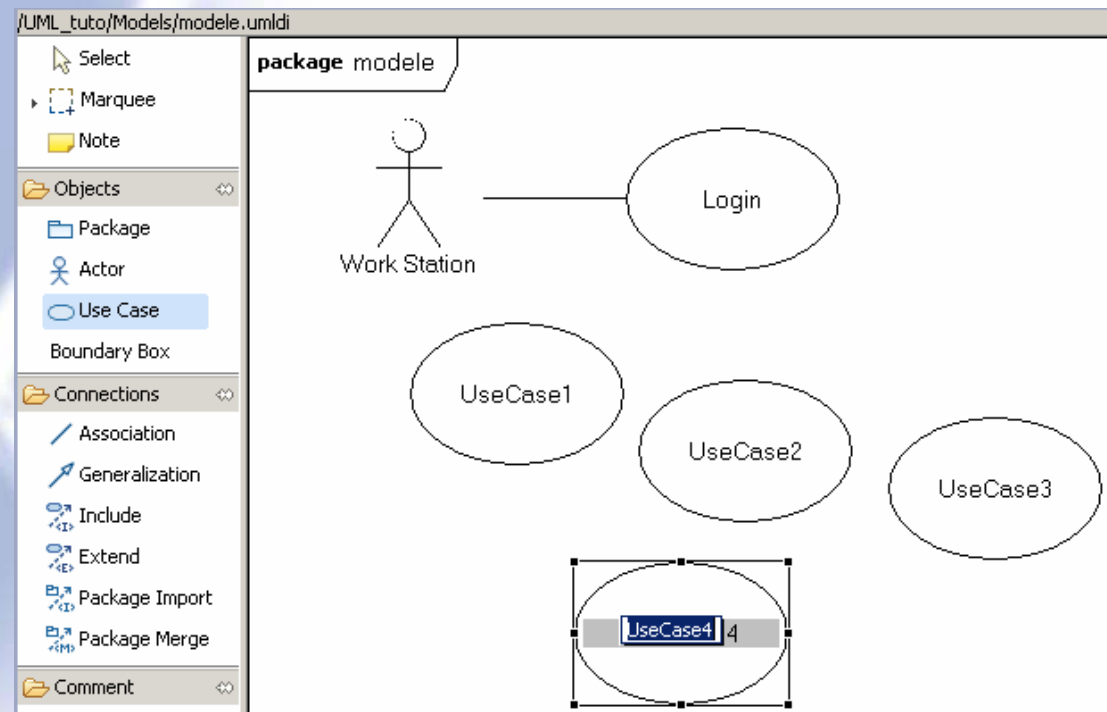




## Edit a diagram (4/8)

**NEW !**

- Multi creation element
  - ▶ You can create several element by press ctrl
  - ▶ Select an element from the palette, press ctrl, click on the diagram as many time as you want to create elements





## Edit a diagram (5/8)

- Properties of a model item (« Properties » view)
  - ▶ Most commonly used properties are in the « model » tab
  - ▶ All the properties are in « advanced » tab

The screenshot shows the 'Model' tab selected in the Properties view. The use case diagram above shows a stick figure labeled 'Maintenance operator' connected to a use case box labeled 'handle proven failure'. The Properties view shows the following fields:

Property	Value
Name:	handle proven failure
Visibility:	public
isAbstract:	<input type="checkbox"/>

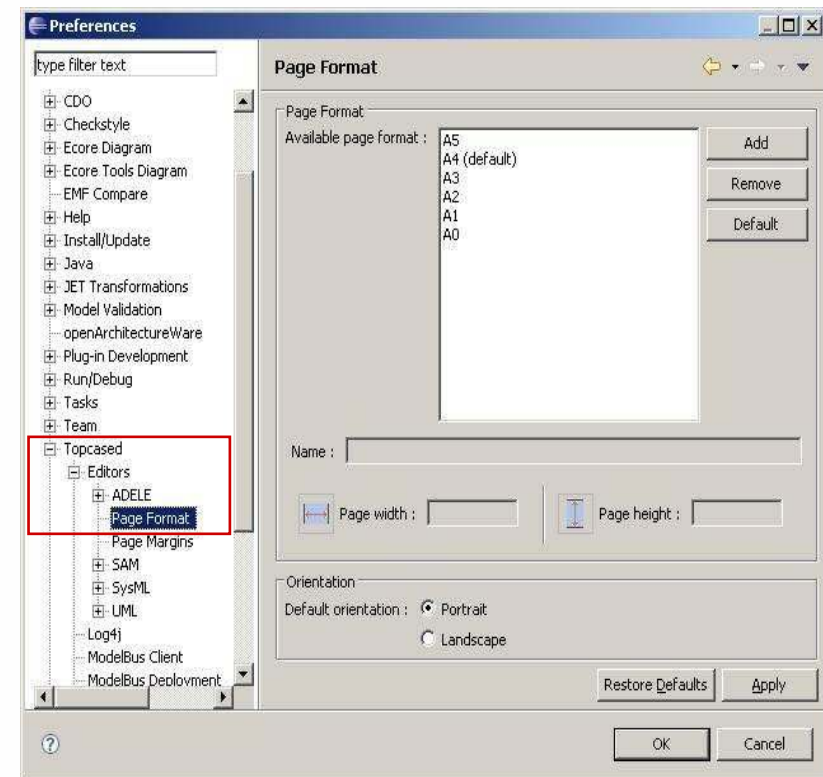
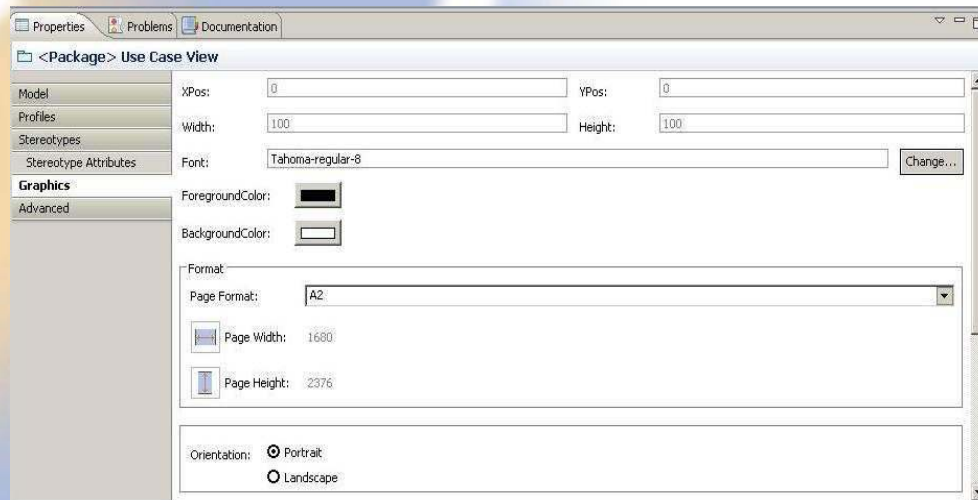
The screenshot shows the 'Advanced' tab selected in the Properties view. The use case diagram above is identical to the previous one. The Properties view shows a table of properties:

Property	Value
UML	
Classifier Behavior	
Client Dependency	
Is Abstract	%% false
Is Leaf	%% false
Name	handle proven failure
PowerType Extent	
Redefined Classifier	
Representation	
Subject	
Template Parameter	
Use Case	
Visibility	Public



## Edit a diagram (6/8)

- Graphical properties linked to the diagram (no item selected)
  - ▶ Colors, font, print format
  - ▶ List of print formats can be modified in the TOPCASED preferences





## Edit a diagram (7/8)

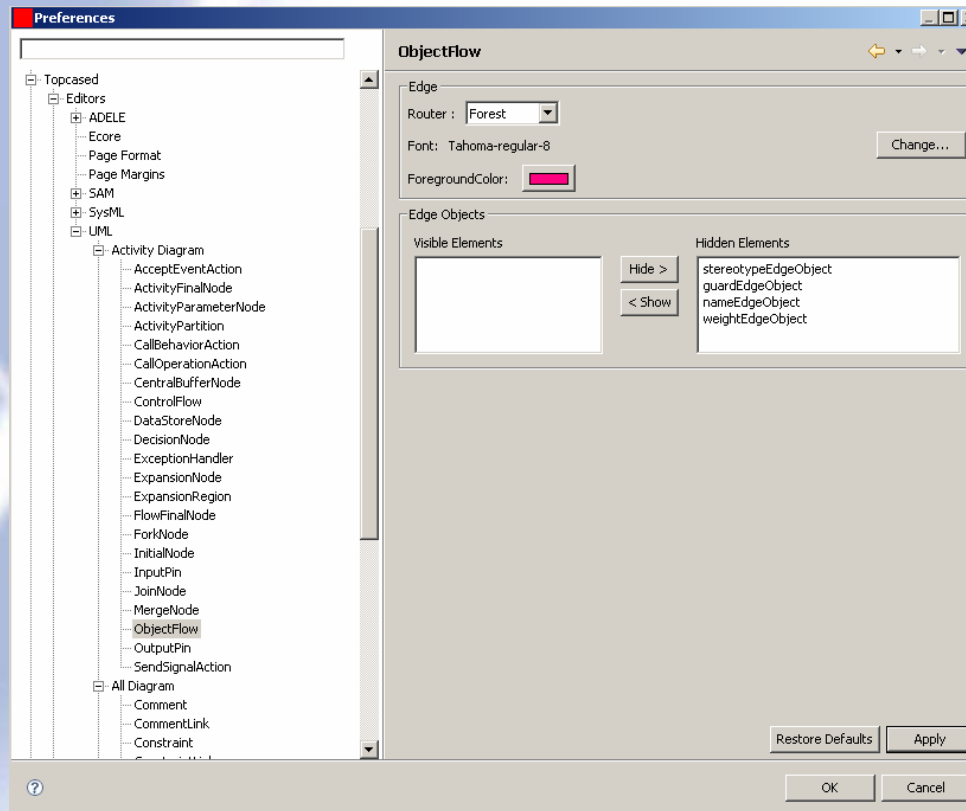
- Graphical properties available for the selected item(s)
  - Adapted to item type (node, connection...)

The screenshot displays the Topcased UML editor interface. At the top, a toolbar contains icons for 'Include', 'Extend', 'Comment', and 'Comment Link'. The main workspace shows a UML diagram with a stick figure actor labeled 'MaintenanceOperator' connected by a blue line to an oval-shaped use case labeled 'handle proven failure'. Below the workspace is a 'Properties' panel for the selected association, titled '<Association> Association1'. The panel includes fields for 'XPos', 'YPos', 'Font' (set to 'Sans-regular-10'), 'ForegroundColor' (set to blue), and 'Router' (set to 'Oblique'). At the bottom of the panel, there are two lists: 'Visible Elements' containing 'Aa stereotypeEdgeObject' and 'Hidden Elements' containing 'Aa srcCountEdgeObject', 'Aa targetCountEdgeObject', and 'Aa nameEdgeObject'. Buttons for 'Hide >' and '< Show' are positioned between the two lists.



## Edit a diagram (8/8)

- Default behavior
  - ▶ Workspace level (preferences) or project level (right click « properties »)
  - ▶ Entirely generic (all languages and all items of the language)





# Document the model elements

- A **note** can be used for diagram only
  - ▶ not stored in the model
- Each model element can be documented through a documentation view
  - ▶ Simple text
  - ▶ Rich Edit text with a dedicated editor
    - Bold, copy/paste...
    - Tables, images, links...
    - Ctrl M or Ctrl Shift M to go next line

Author : Raphael FAUDOU  
Date : 2008 October the 9th

afficher une alerte

créer une alerte

Document

Comments | Resources

• **Voici les caractéristiques exigées pour l'affichage des alertes :**

caracteristiques

Exigence	conformité
Sous forme arborescence	obligatoire
alignées à gauche	optionnel

Normal Times New Roman 3

• **Voici les caractéristiques exigées pour l'affichage des alertes :**

caracteristiques

Exigence	conformité
Sous forme arborescence	obligatoire
alignées à gauche	optionnel

OK Cancel

Document

Comments | Resources

• **Voici les caractéristiques exigées pour l'affichage des alertes :**

caracteristiques

Exigence	conformité
Sous forme arborescence	obligatoire
alignées à gauche	optionnel

Informations Listes Passages Aperçu PGT Rechercher

Missions

Mission sélectionnée : SP1

Etat :

Afficher les éléments

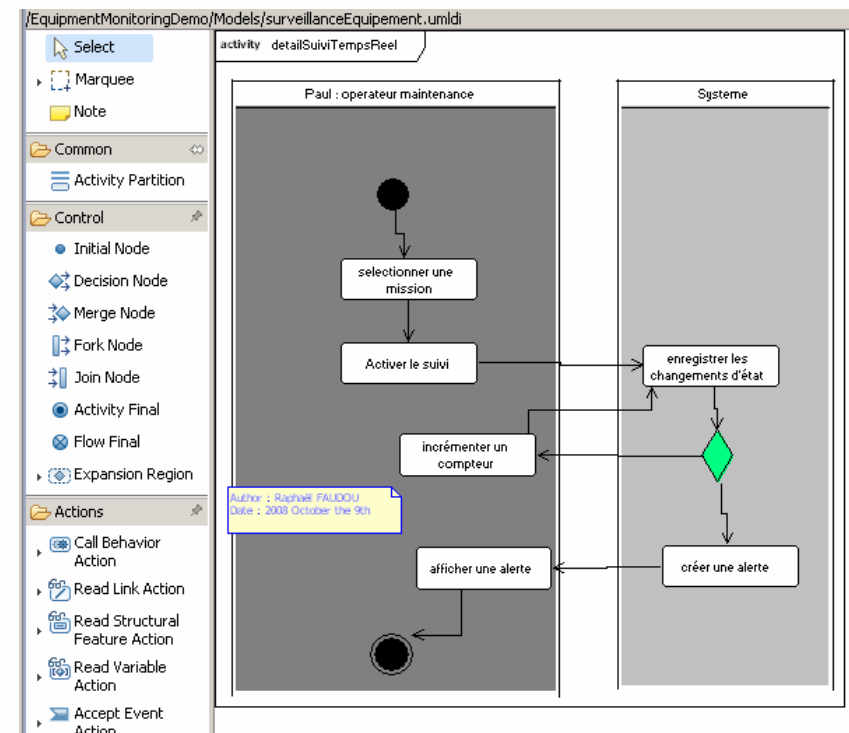
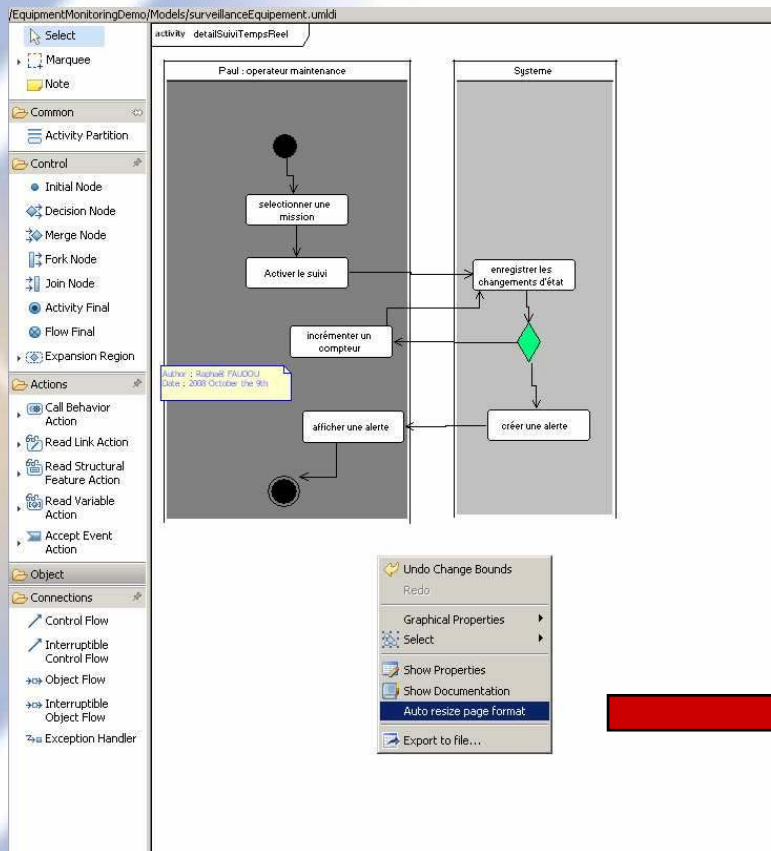
Tâches de fond	Etat	Non
Procédures manuelles		

Edit



# Autoresize the diagram

- Before, some parts of the A4 diagram format are blank...



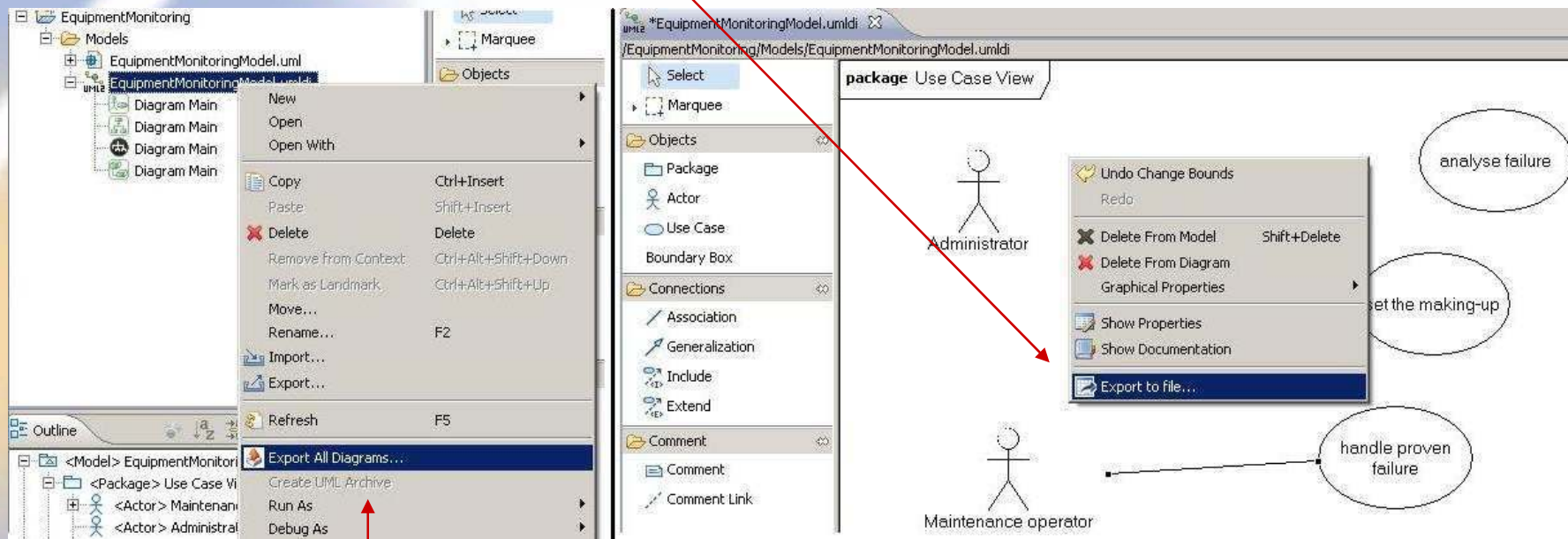
- After, the diagram format has been adapted to its content





# Export the diagram as an image

- Export the current diagram
  - ▶ Contextual menu « export file » then select the directory and the format



- Export all the diagrams
  - ▶ Select xxx.uml-di file
  - ▶ Contextual menu « export All Diagrams... » (then select the directory and the format)



## Add a diagram (1/3)

- Select a package from Outline view
  - ▶ For instance the model (root package)
- Add (create) a diagram via the contextual menu
- Change the name in the properties view
  - ▶ For instance « overview »

Property	Value
Diagram Link	
Name	overview
Position	0,0
Reference	
Size	100,100
Viewport	0,0
Visible	true
Zoom	1.0

- Create child
- Add diagram
- Generate Primitive Types
- Import Primitive Types
- Apply Profile
- Unapply Profile
- Apply Stereotype
- Unapply Stereotype
- Delete From Model
- GOF Patterns
- Load Resource...
- Remove UUID Annotations

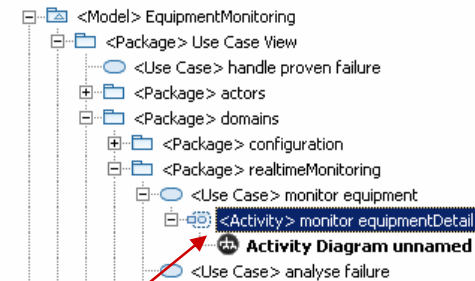
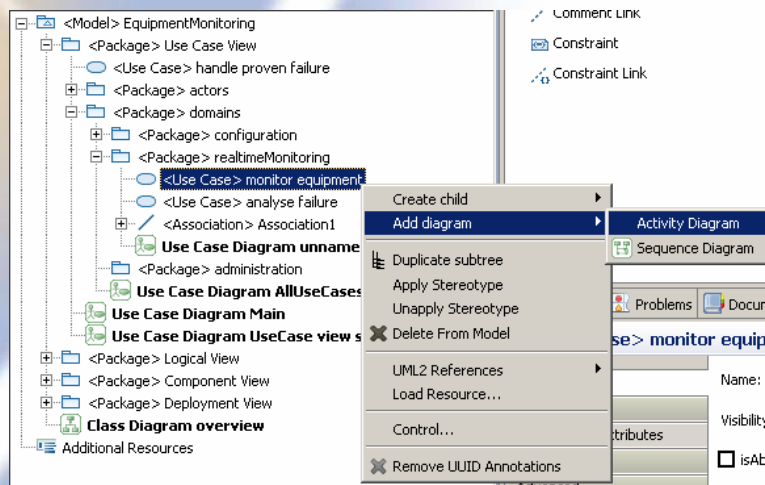
- Class Diagram
- Deployment Diagram
- Sequence Diagram
- State Machine Diagram
- Use Case Diagram

- Edit the diagram with the palette
- Or drag existing items from the outline view...  
...to drop them on the new diagram
  - ▶ Example : the main packages (UC view,...)
- Remark : an UML item decorated with green arrow means it contains sub-diagrams



## Add a diagram (2/3)

- Remark : some diagrams cannot be directly created in a package
  - ▶ The UML metamodel imposes the creation of another UML item beforehand
    - « activity » to create an activity diagram
    - « class » to create a composite structure diagram
    - « interaction » for sequence diagram,
    - « stateMachine » for state diagram
- Example : description of a use case through an activity diagram
  - ▶ Select the use case “monitor equipment” in the outline
  - ▶ Add an activity diagram with the contextual menu

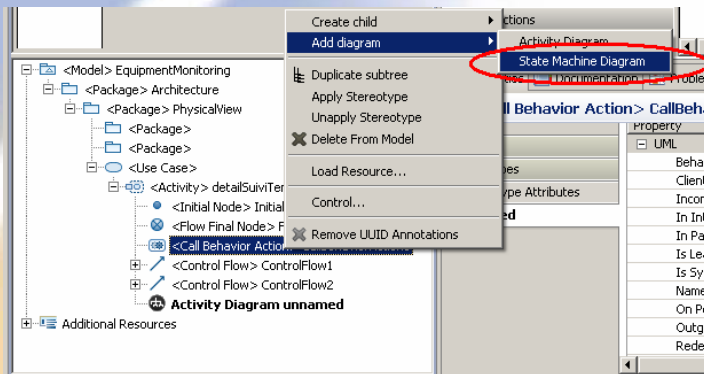


- ▶ An activity has been created below the use case

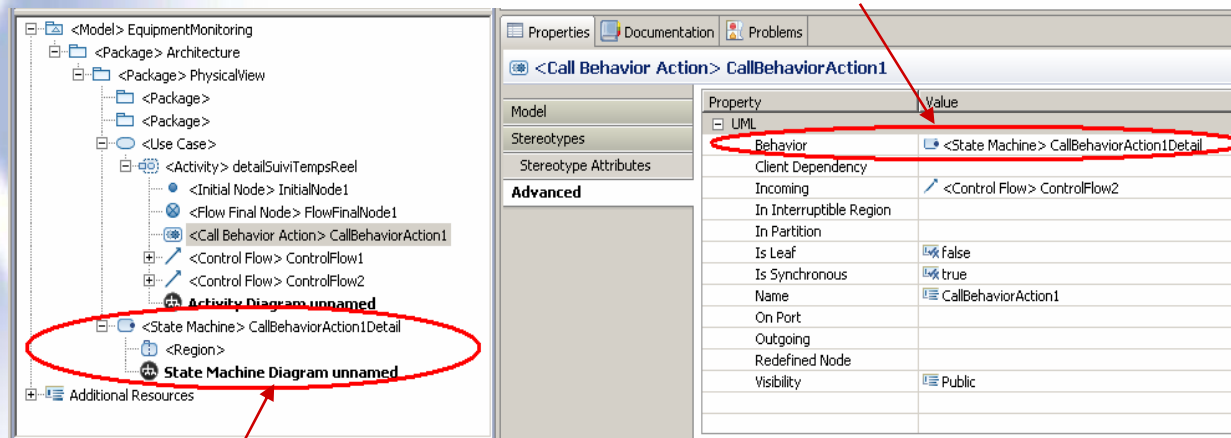


## Add a diagram (3/3)

- Other shortcuts : to add an activity diagram or a state machine diagram to a CallBehaviorAction



- Automatic **association** to current diagram

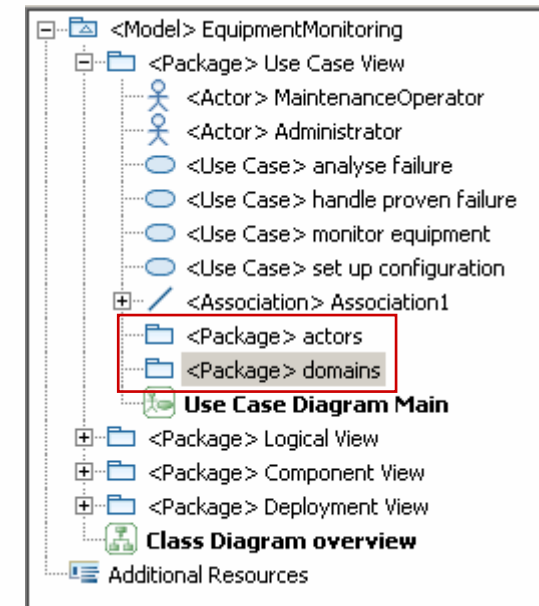
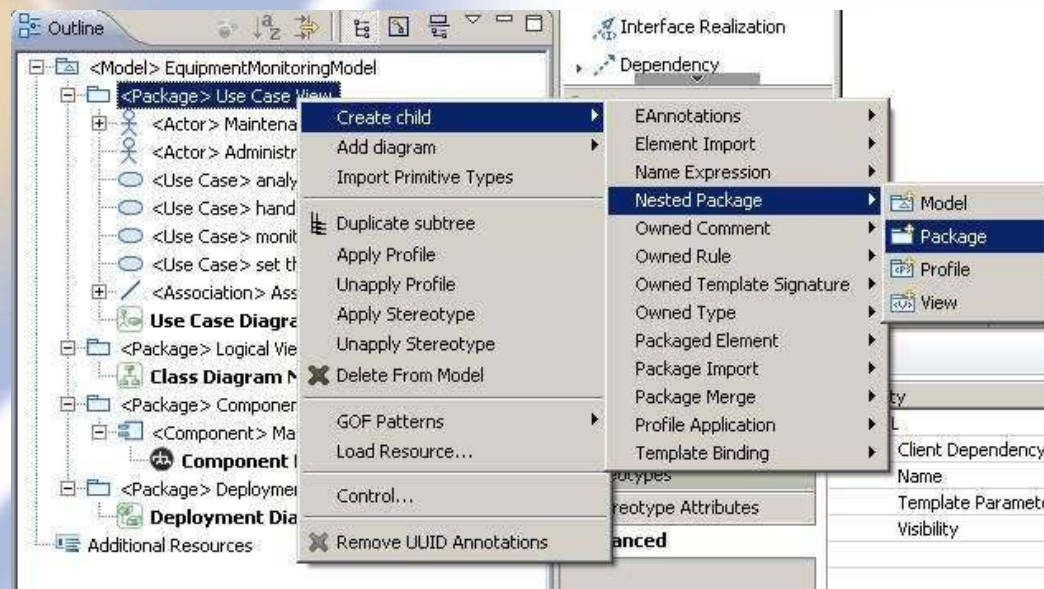


- Automatic creation of the **new diagram**



## Adjust the structure of the model (1/2)

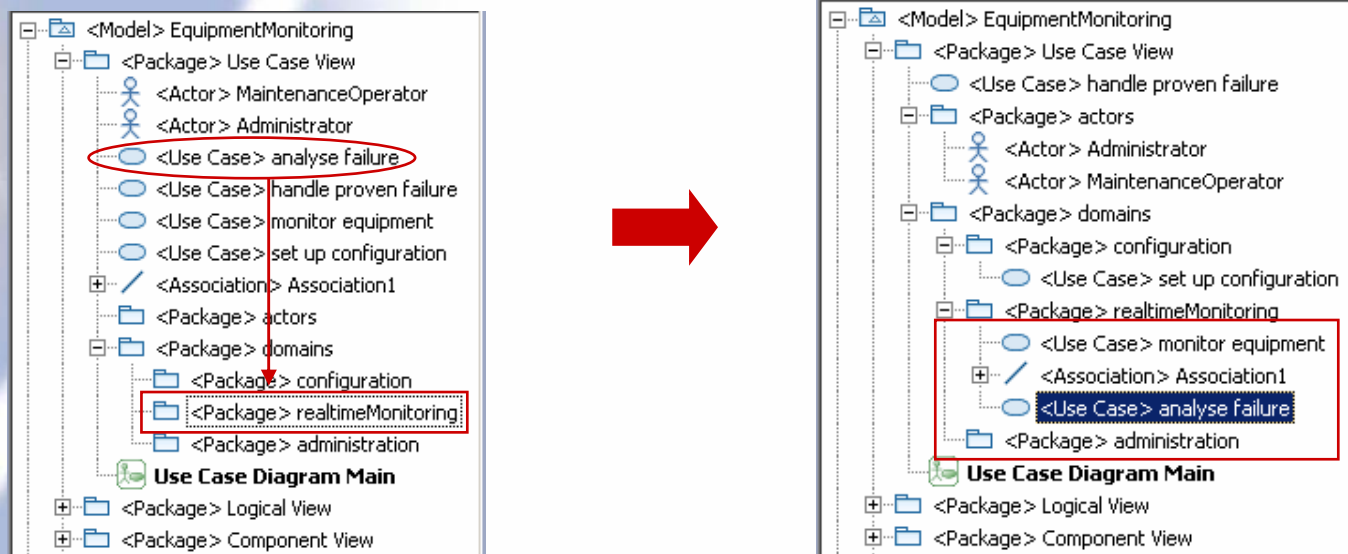
- Requirement to adapt the initial template
  - ▶ Create sub-systems, functional domains...
- Create a package
  - ▶ Either from outline view (CreateChild contextual menu)
  - ▶ Or by the palette in the class diagram or use case diagram
  - ▶ Example : separate actors and use cases in the use case view
    - Create package “actors” and package “domains”





## Adjust the structure of the model (2/2)

- Copy or move a part of the model
  - Drag-and-drop or make a duplicate of a sub-tree in the outline
- Example : Move the actors to “actors” package and UC to functional domains



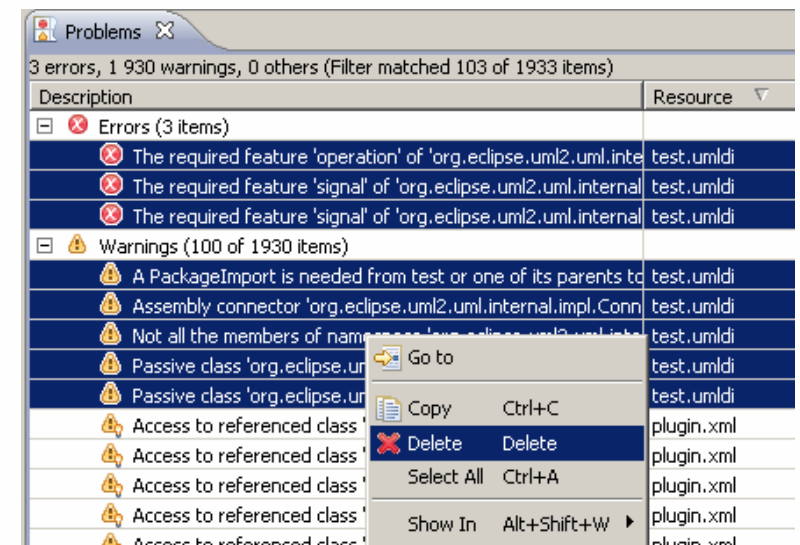
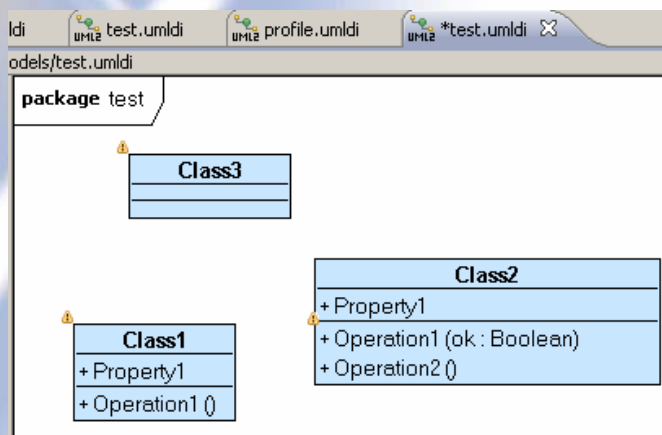
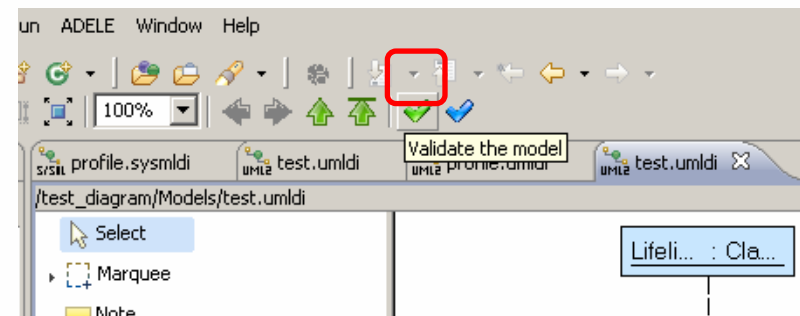
- Remark: duplicating a sub-tree of the model duplicate the diagrams



# Validate the model

**NEW !**

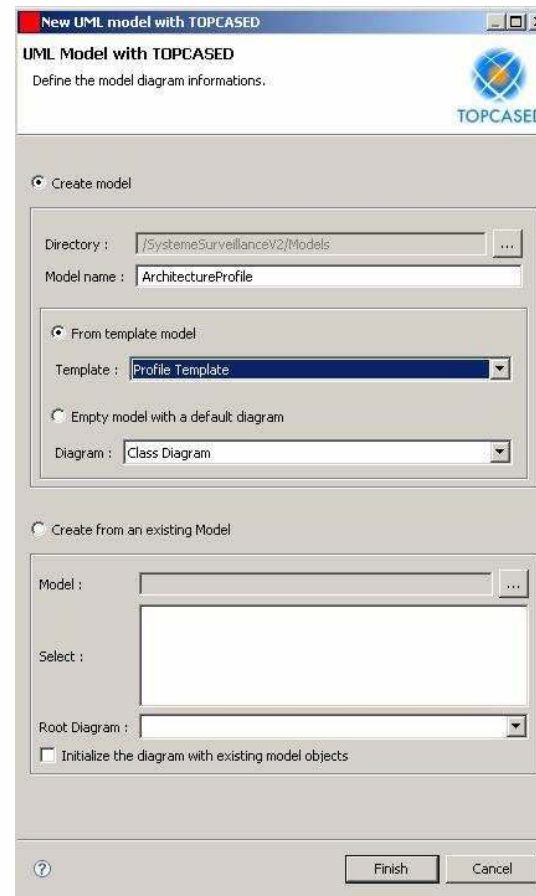
- Topcased validation check if your model is conform to the UML specification
- Click on the validate button in the tool bar
- Errors and warnings are shown in the problems view and on elements in diagrams and outline view
- To remove the warnings just delete them in the problem view





## Specialization of items – UML profile (1/7)

- Create a new UML model based on the « Profile » template
- Example : « architectureProfile »

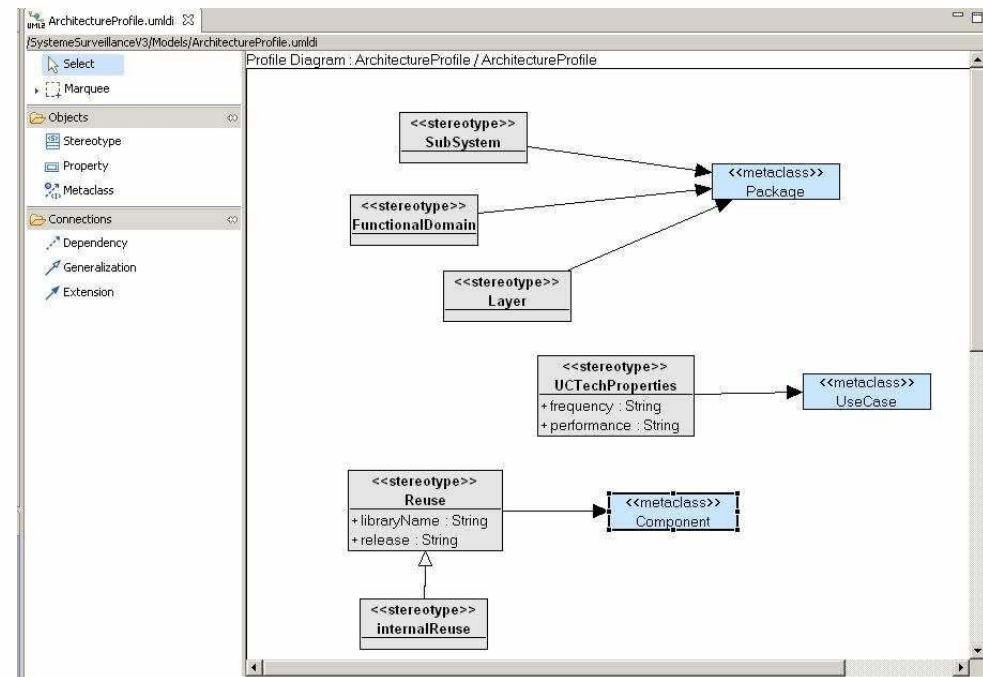






## Specialization of items – UML profile (2/7)

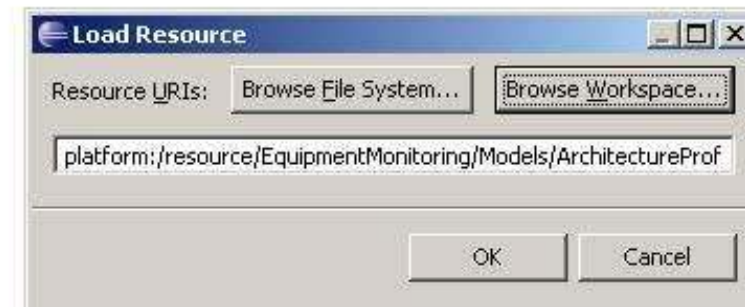
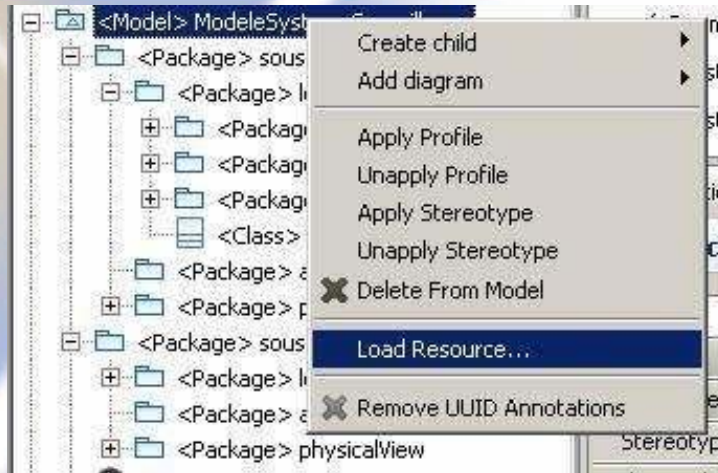
- Create stereotypes and indicate the items to which they are applicable
  - ▶ Relation with meta class
- Possibility of describing the properties of a stereotype
- Possibility of inheriting between stereotypes
- Save the profile
  - ▶ Save the profile
  - ▶ Accept “Define the profile before saving”





## Specialization of items – UML profile (3/7)

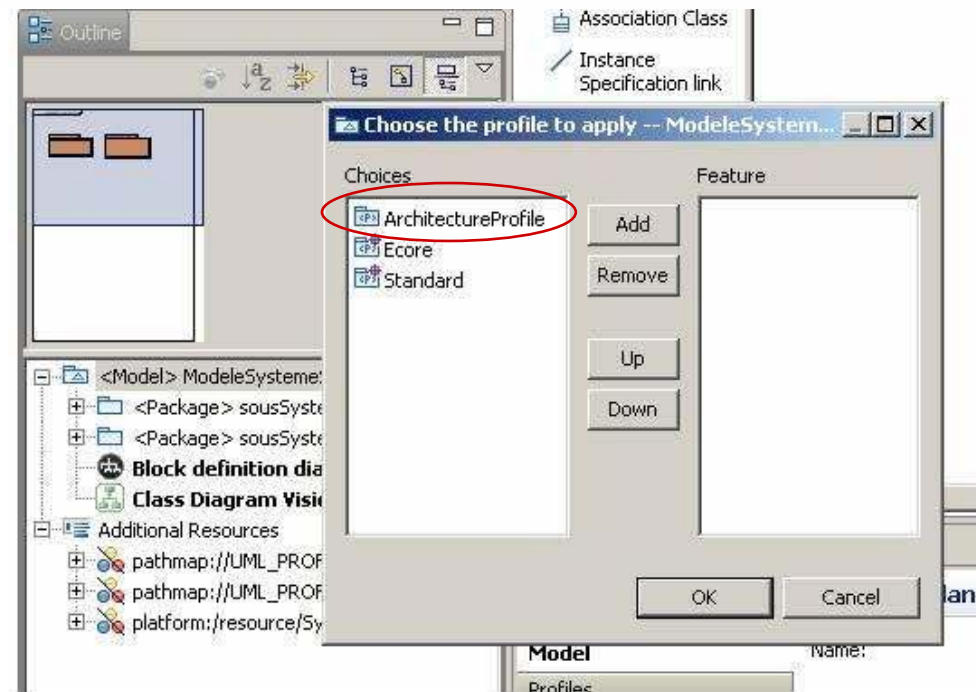
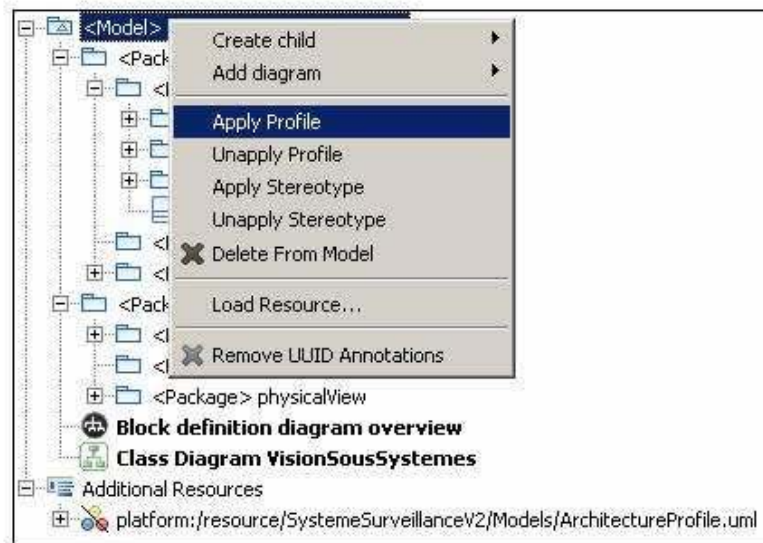
- Referencing the profile
  - ▶ Load the model of the profile on the main model
    - Select the model from the « outline » view
    - Contextual menu « load resource »
    - Select the profile model (.uml extension)





## Specialization of items – UML profile (4/7)

- Application of the profile to the model
  - ▶ Select the model from « outline » view and from « apply profile »
    - The “ArchitectureProfile” profile is then available





## Specialization of items – UML profile (5/7)

- Select an item in the diagram
  - ▶ Select the stereotype from the properties view and apply it
  - ▶ The specific properties of the stereotype are then available

The screenshot illustrates the process of applying a stereotype to a use case in the UML editor. On the left, a package diagram shows several use cases: 'failures processing', 'real-time equipment monitoring', 'equipment administration', and 'system configuration'. The 'real-time equipment monitoring' use case is selected, and a red arrow points to the 'UC Tech Properties' stereotype in the Properties view. The Properties view shows the 'UC Tech Properties' stereotype with the following attributes:

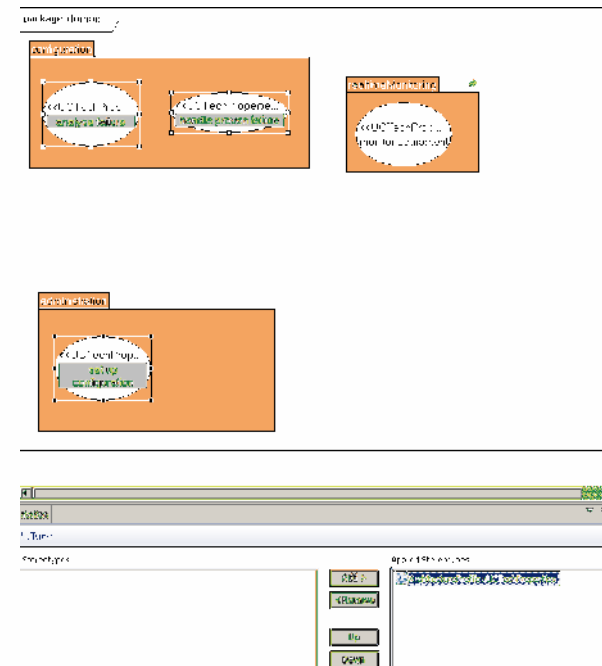
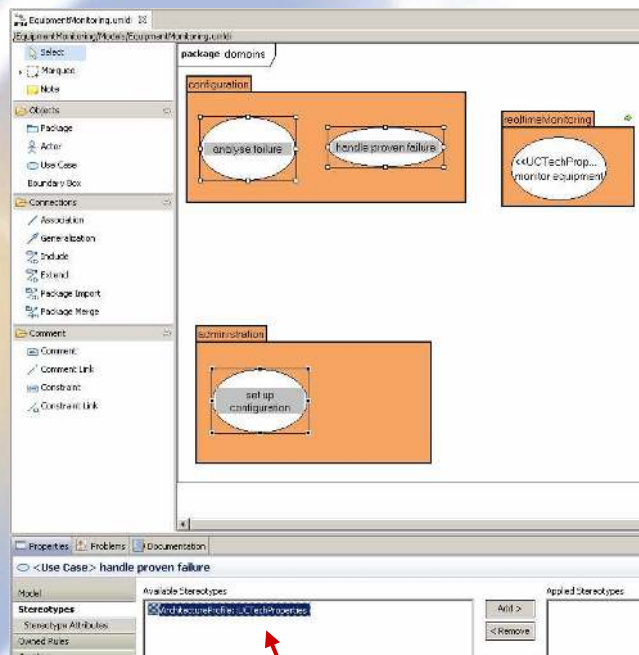
Property	Value
UC Tech Properties	
Frequency	200 messages per second
Performance	Max 50% CPU

On the right, the 'equipment administration' use case is shown with the 'UC Tech Properties' stereotype applied, and the Properties view shows the same attributes. A red arrow points from the 'UC Tech Properties' stereotype in the Properties view to the 'UC Tech Properties' stereotype in the Properties view of the 'equipment administration' use case.



## Specialization of items – UML profile (6/7)

- You can apply a stereotype to several elements
  - Select several use cases using the Control key
  - Select the stereotype from the properties view and apply it



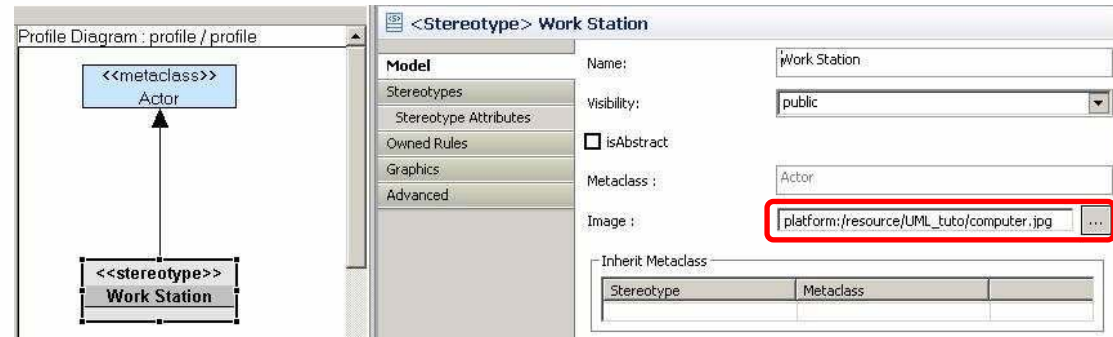
Note that the available stereotypes are filtered depending on the selected element types.



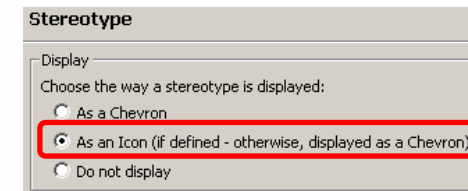
# Specialization of items – UML profile (7/7)

**NEW !**

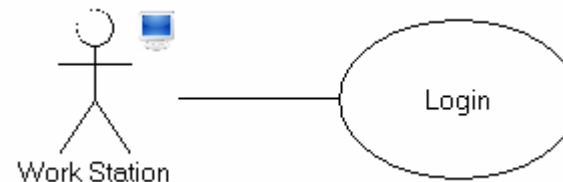
- Customize figure's element using stereotype
  - ▶ Define a stereotype and add an image



- ▶ Change stereotypes preferences : select “As an Icon” in Window > Preferences > Topcased > UML/SysML > All Diagrams > Stereotype



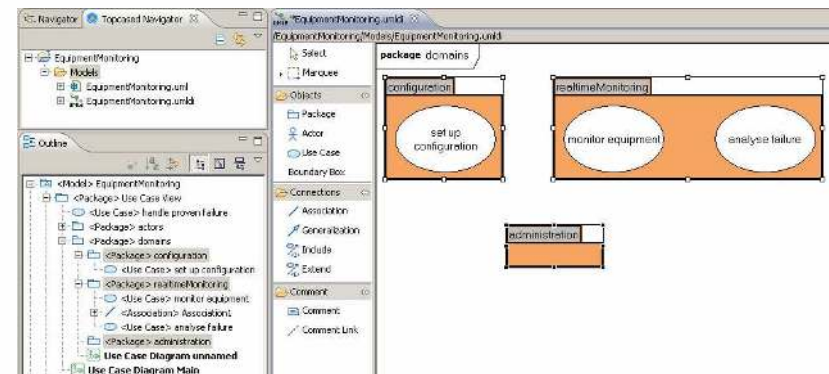
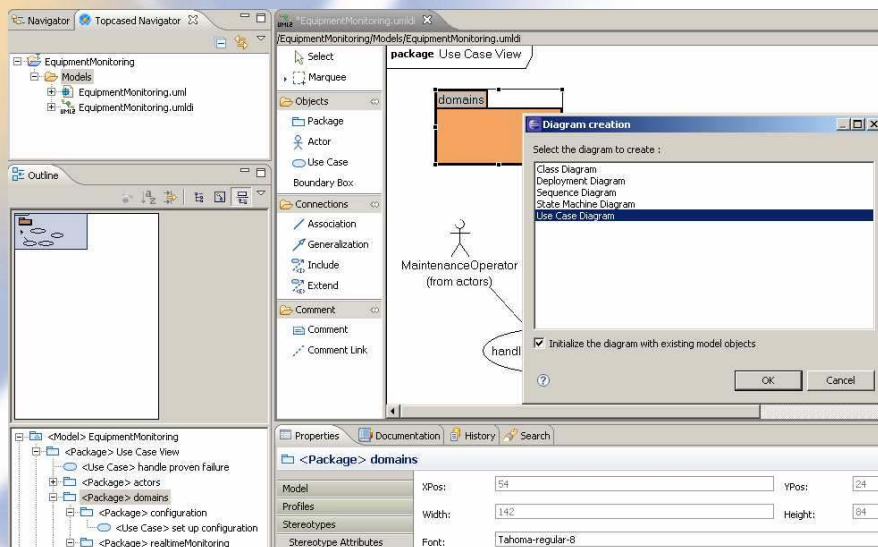
- ▶ Apply the stereotype on element.






## Navigation in the diagrams (1/5)

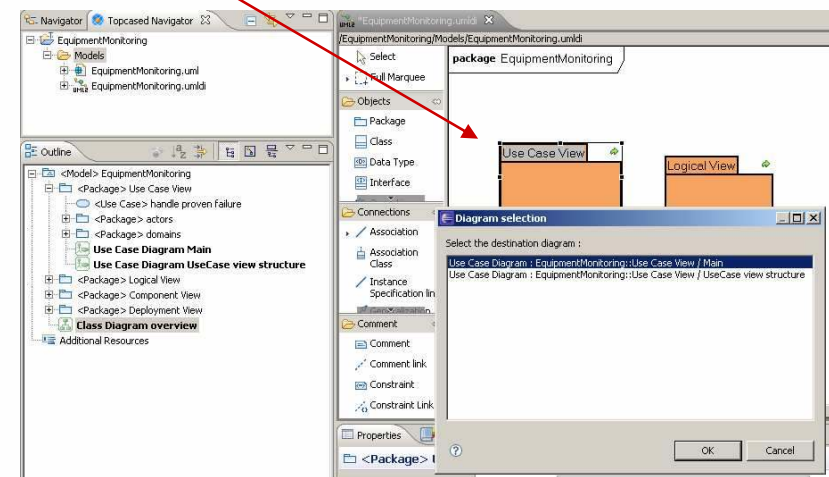
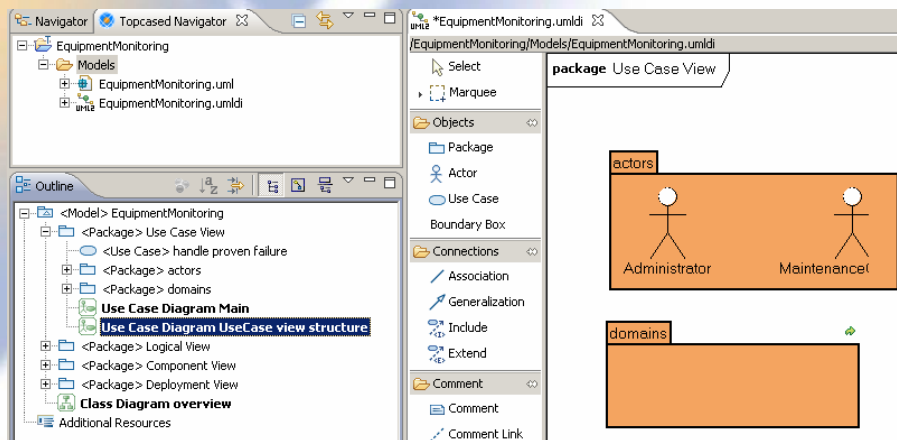
- Arrow cursor to navigate in the diagrams
  - ▶ Previous, next, parent, main (root) diagram
- Some UML items (containers) allow navigation to children diagrams
  - ▶ Double click on the item (package, class...)
  - ▶ A creation wizard is displayed if no child diagram is defined
  - ▶ Example : let us drag and drop the “domains” package in the UC diagram
    - Then, let us double click on the “domains” package in the diagram
    - A wizard is displayed : let us choose “use case diagram”...





## Navigation in the diagrams (2/5)

- Double click on an item containing several children diagrams
  - ▶ A list of diagrams is displayed to the user so that he can make a choice
  - ▶ Example
    1. Create a new UC diagram in the use case view
    2. Drag & drop “actors” and “domains” packages
    3. Select now the “overview diagram” by clicking the “root diagram” arrow 
    4. Double click on the “use case view” ...  
...A list of two diagrams is displayed

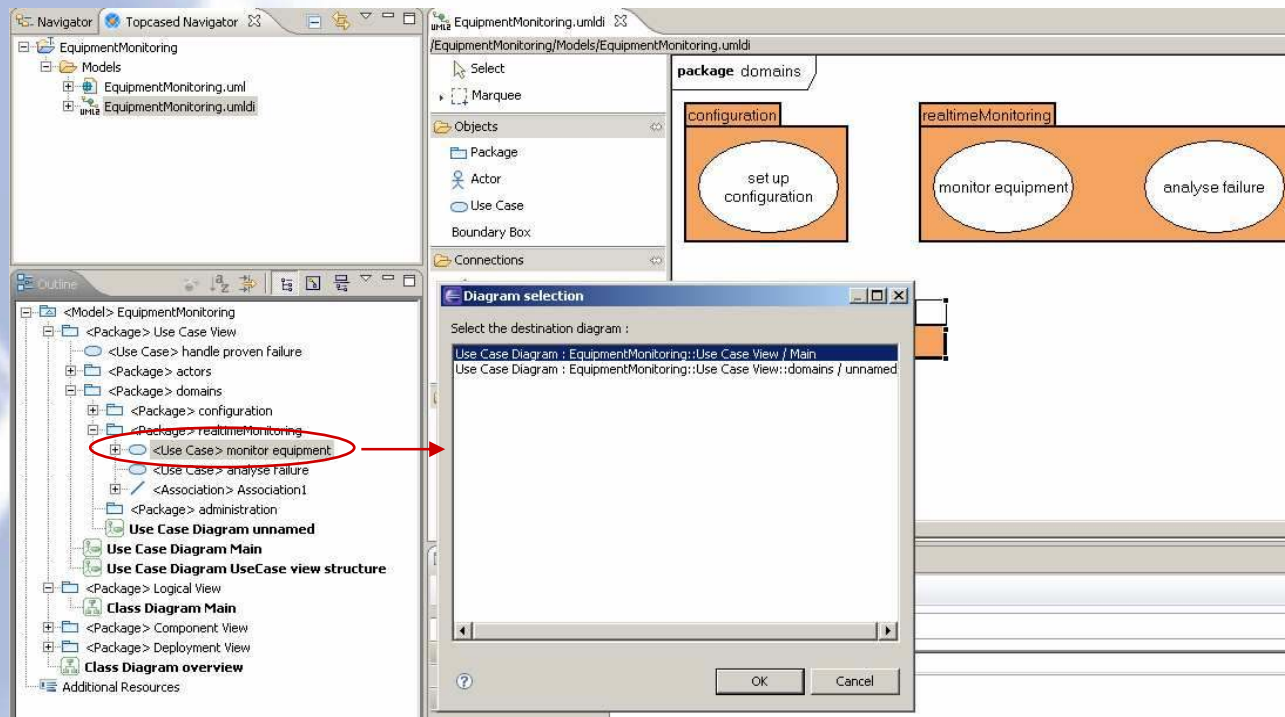






## Navigation in the diagrams (3/5)

- Navigation from a model item in outline view (double click)
  - ▶ If the item is present in a single diagram, this diagram is open
  - ▶ If the item is present in several diagrams, a list is displayed
  - ▶ Example : double click on “monitor equipment” use case

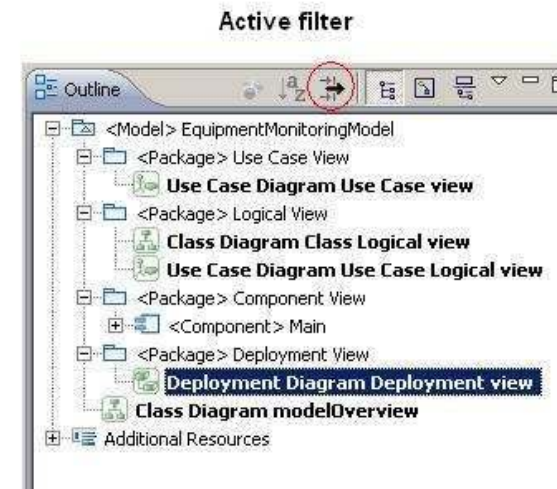
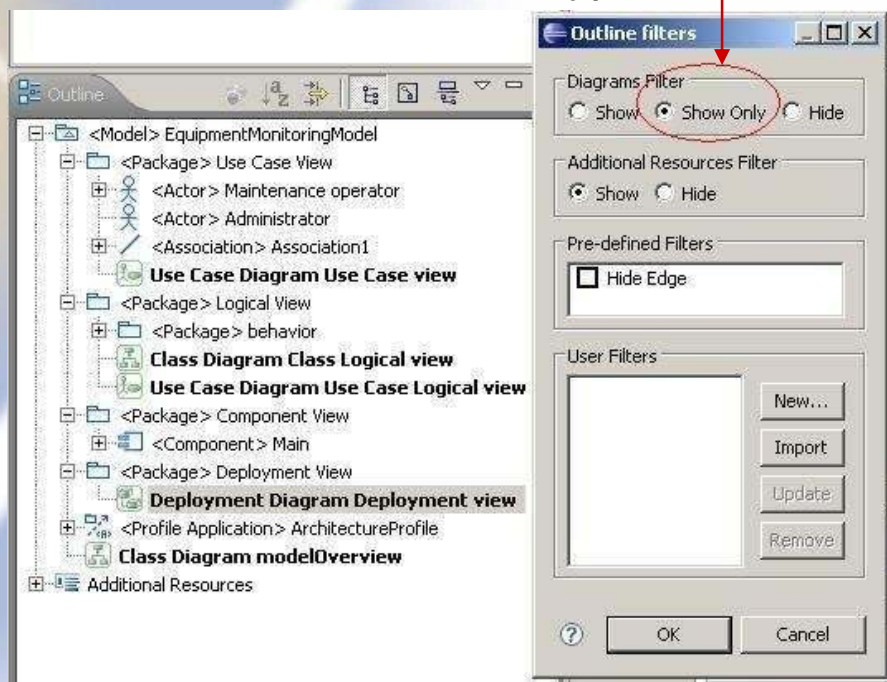
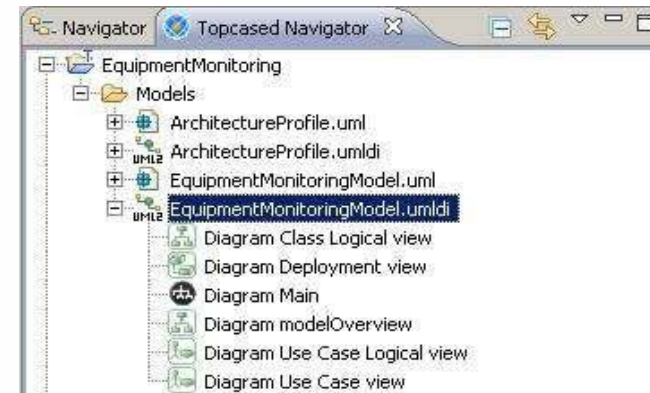


- ▶ If the item is not present in any diagram, double click is inactive



## Navigation in the diagrams (4/5)

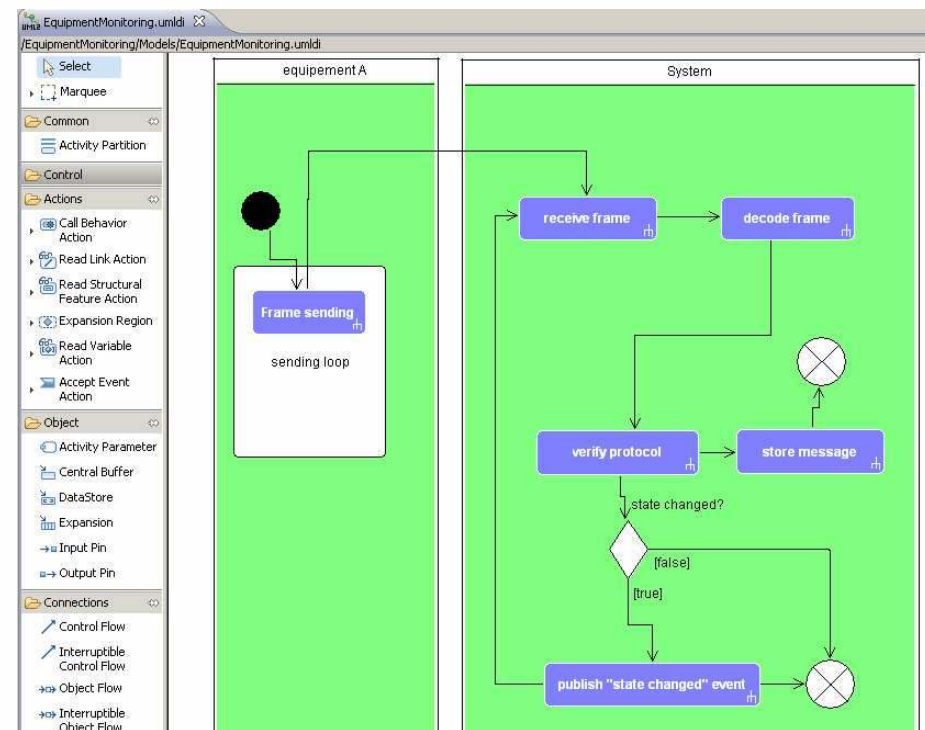
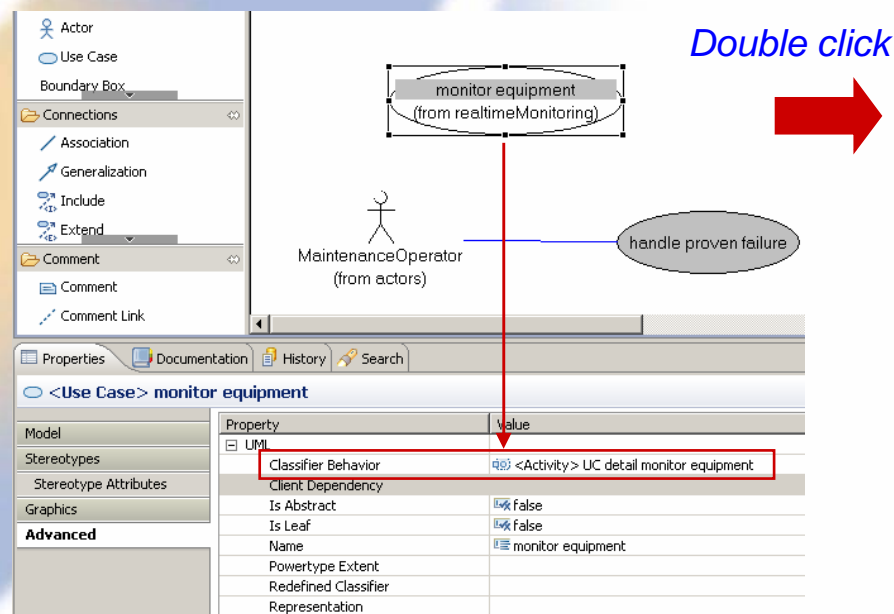
- Navigation from TOPCASED navigator view
  - ▶ Diagrams are shown by name and are flat
- Navigation from Outline view
  - ▶ Possibility of filtering only the diagrams
  - ▶ Or create filters on UML types





## Navigation in the diagrams (5/5)

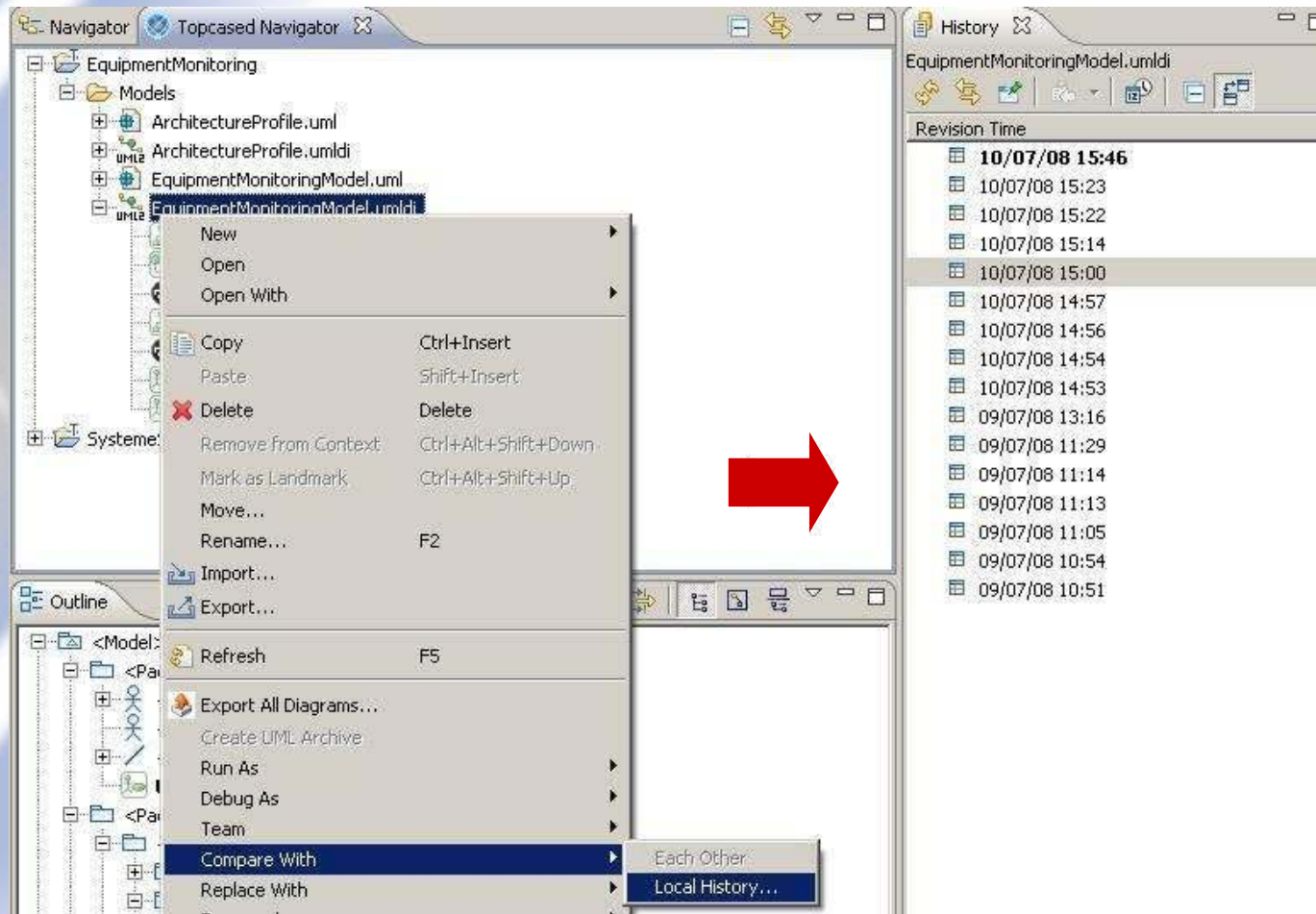
- Navigation from a use case
  - ▶ A use case does not have sub-diagrams (not allowed by UML metamodel)
  - ▶ However navigation is possible if there is a diagram linked to UC behavior
  - ▶ Example : activity behavior for UC “monitor equipment”





## Version management of models (1/5)

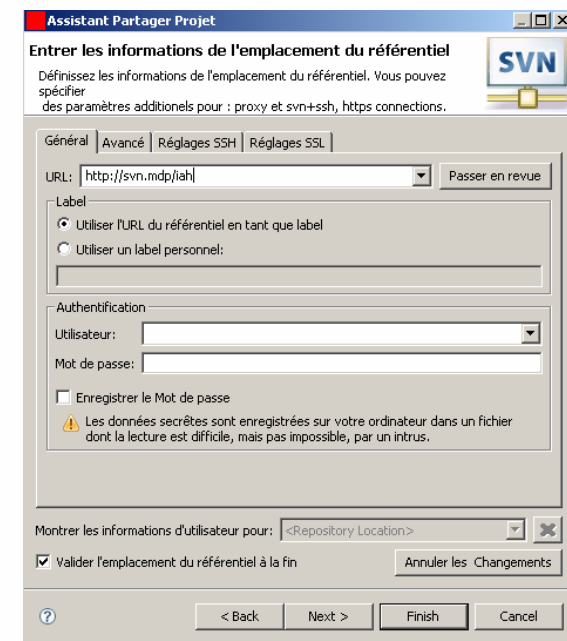
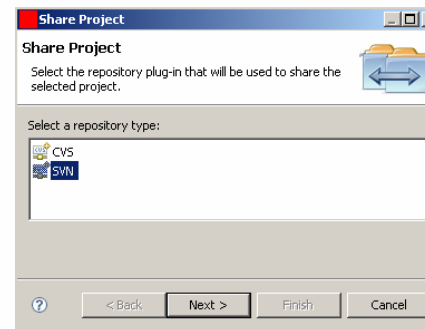
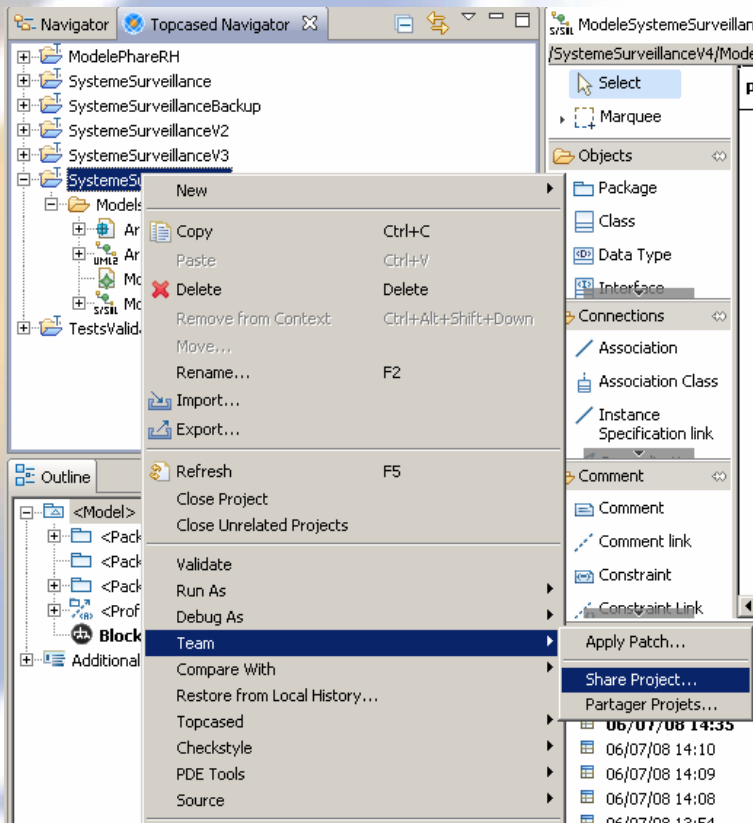
- Local history provided by Eclipse





## Version management of models (2/5)

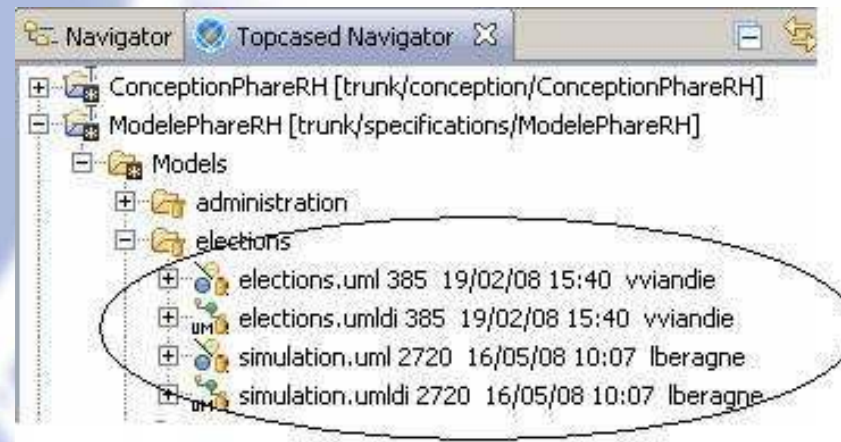
- Connect to a version management system
  - ▶ Share the project (open the installed connectors)
  - ▶ Possibility of using or creating a repository





## Version management of models (3/5)

- The project is thus decorated as it was connected to the repository
  - ▶ URL of the branch of the repository indicated in the project
  - ▶ Icon for each resource linked to version manager
    - Information on the last revision (author, date...)

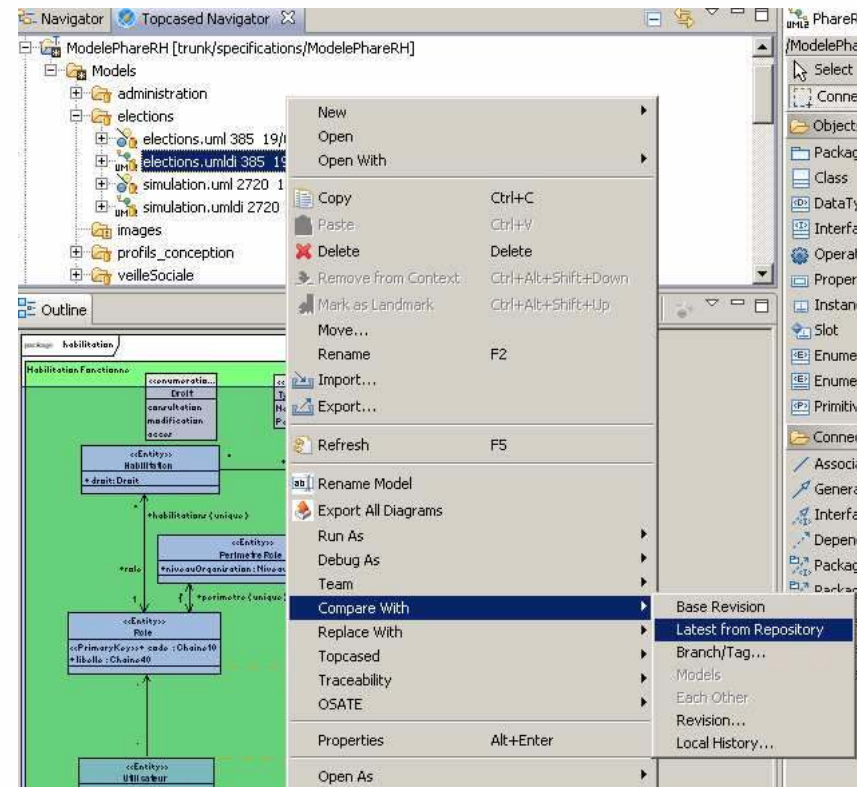
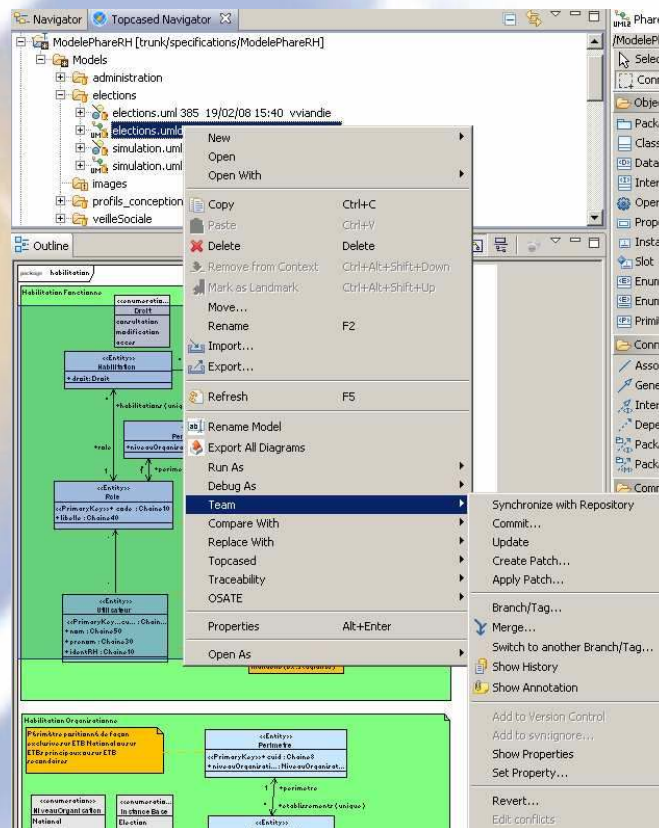


- Perspective dedicated to each version management
  - ▶ Management of repositories, history (revisions), synchronization, comparison...



## Version management of models (4/5)

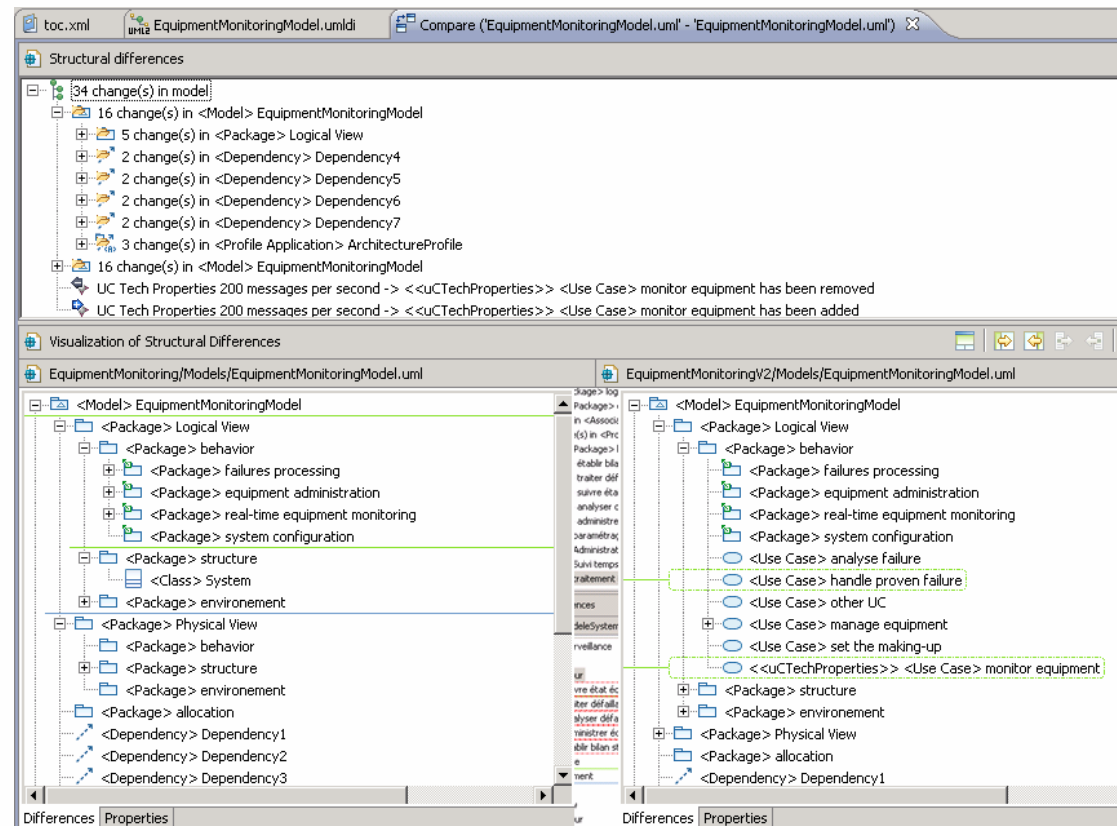
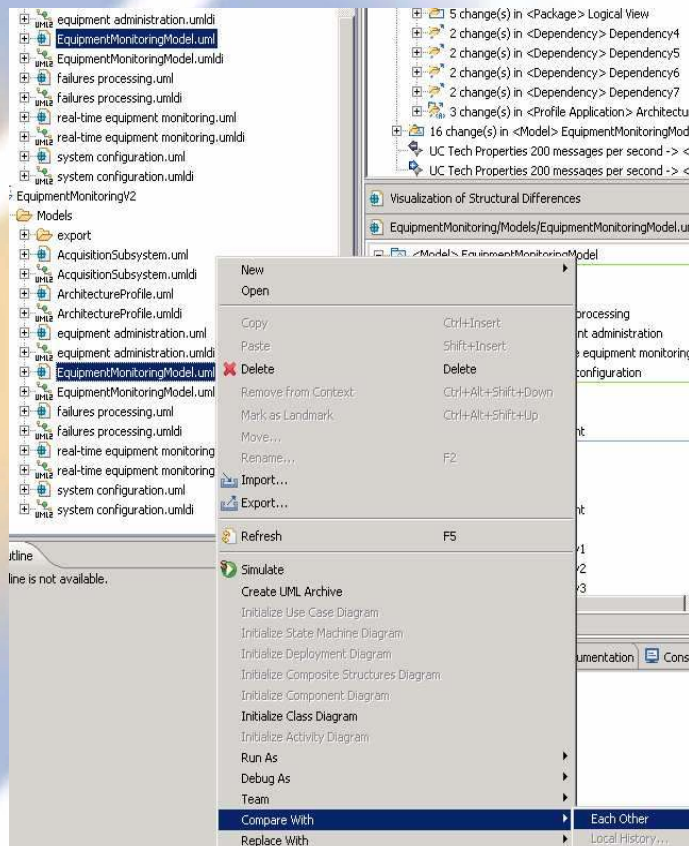
- Operations of version management available for each model
  - ▶ Contextual menu « Team » : Synchronize, commit, update, ...
  - ▶ Contextual menus « Compare with » and « Replace with » : revision...
    - Remark : compare « text » type





# Version management of models (5/5)

- Compare models
  - ▶ Integrate « EMFCompare » plugin (from Eclipse EMF Tools)

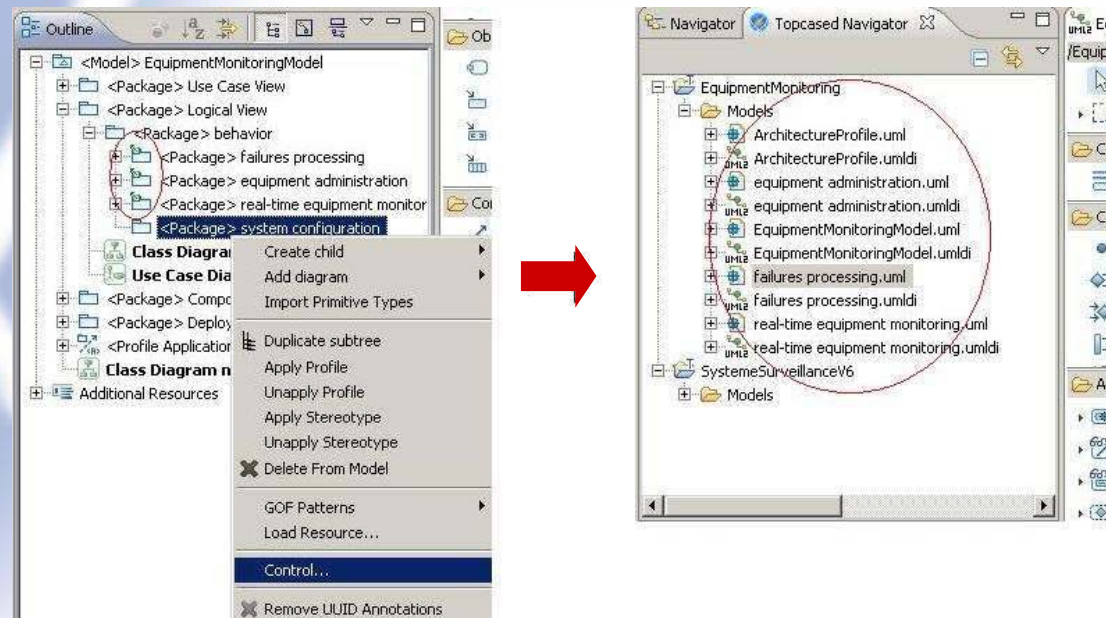






## Collaborative management of models (1/3)

- Goal : it must be possible for several people to work on the same model
  - ▶ If only one file, modifications imply merging → difficult, risk of data overridden
  - ▶ Better approach : divide the model into several resources
- “Control” command available on any container item of the model (package, class...)
  - ▶ Remark : saving is required to finalize the creation of sub-model
  - ▶ Example : divide according to the functional domains

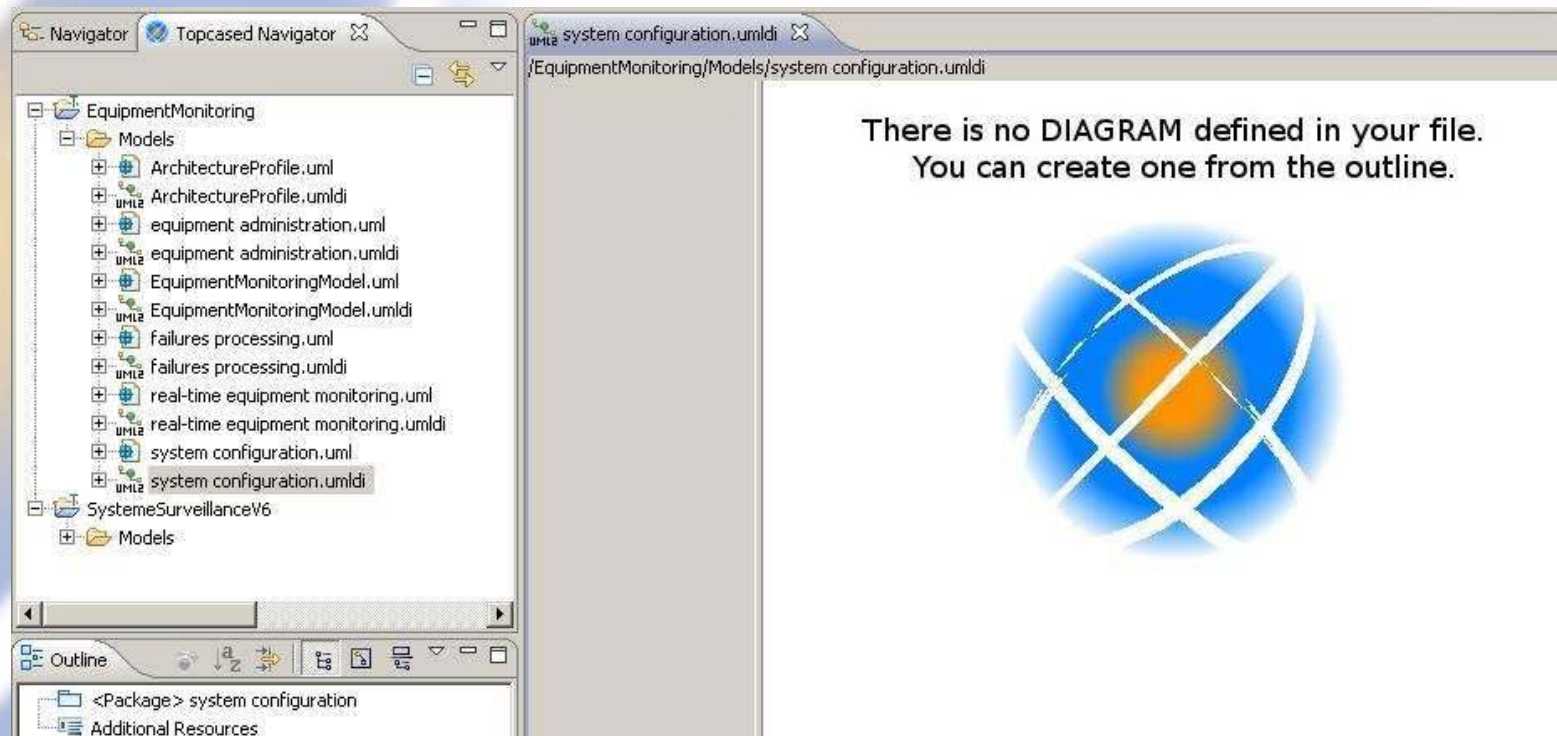


- ▶ A sub-model is created, referred to by the parent model (specific icon)



## Collaborative management of models (2/3)

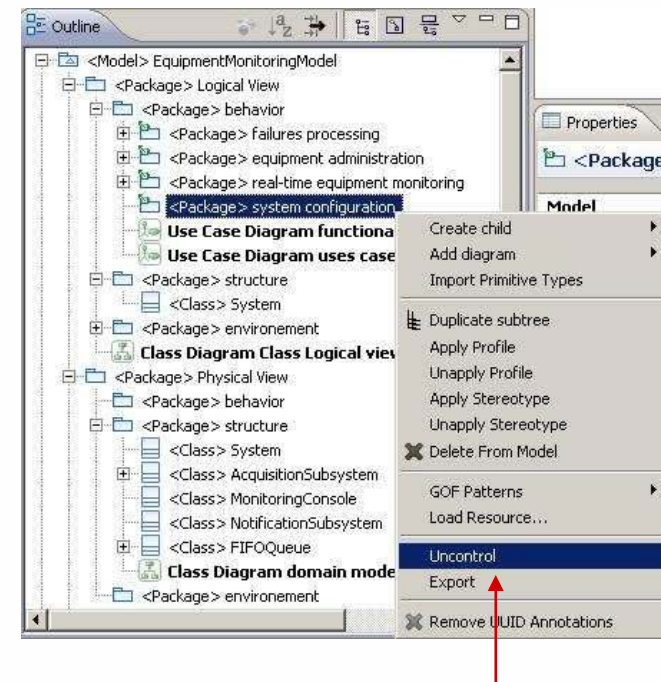
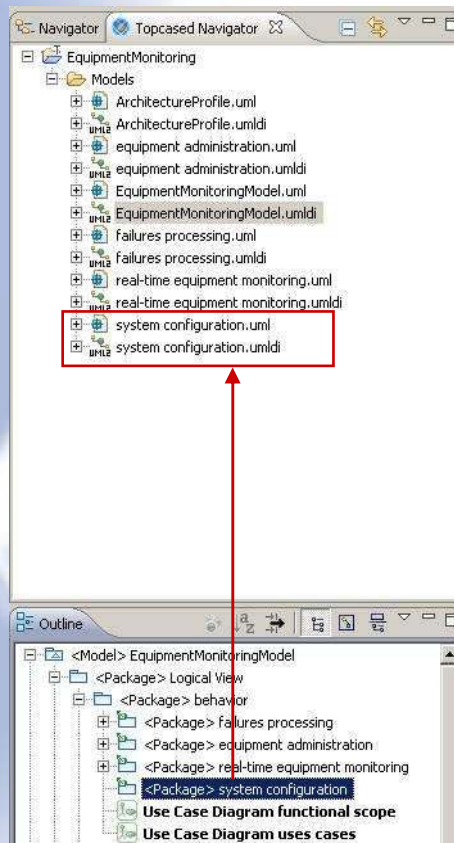
- Each sub model is a model
  - ▶ It can thus be opened by TOPCASED (double click on file xxx.uml*di*)
  - ▶ So far as it does not have a diagram, a message is displayed
  - ▶ It is possible to create one from the outline as in the case of any model





## Collaborative management of models (3/3)

- The sub-model can be seen from the parent model
  - ▶ Its file is referred to by the main model's file

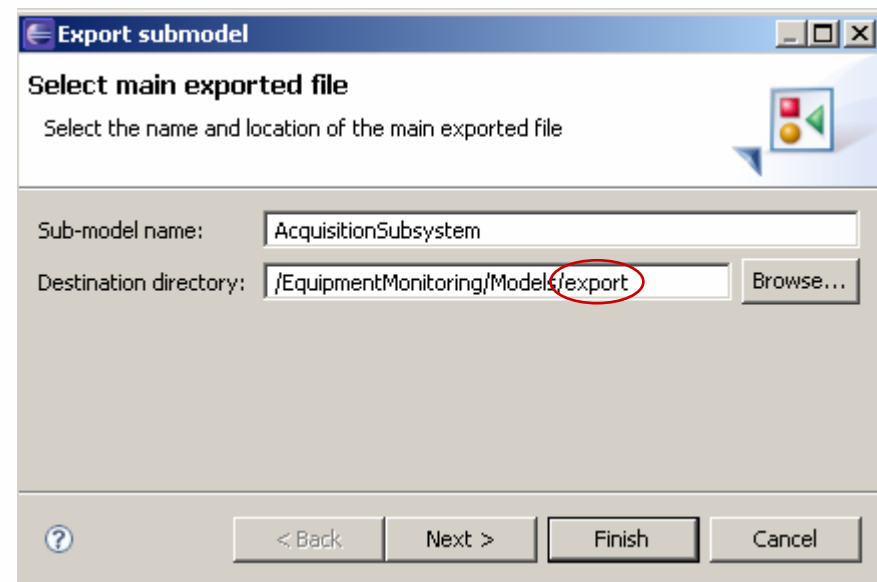
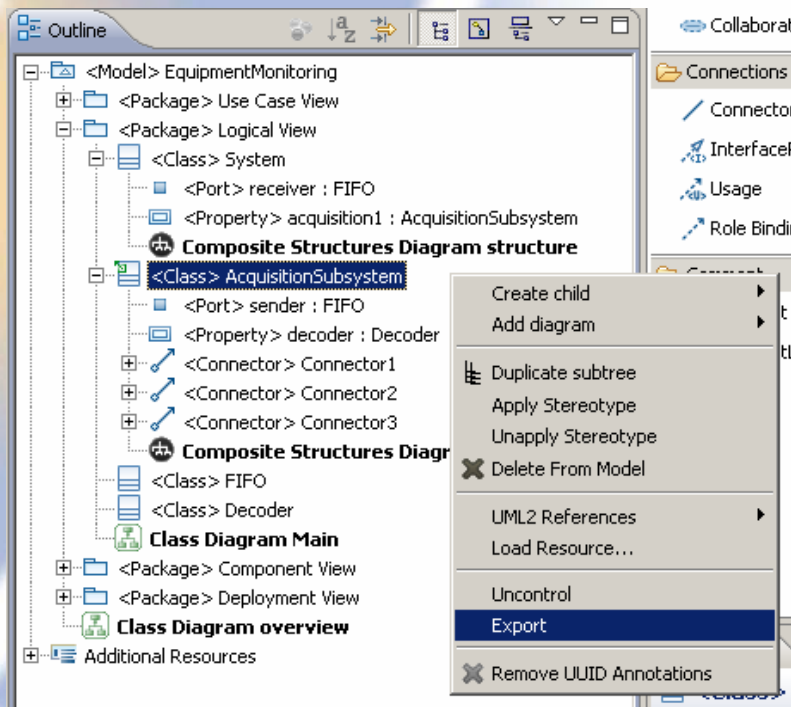


- The sub-model can be included again in the parent model through “uncontrol”



## Work in « out-sourcing » mode (1/8)

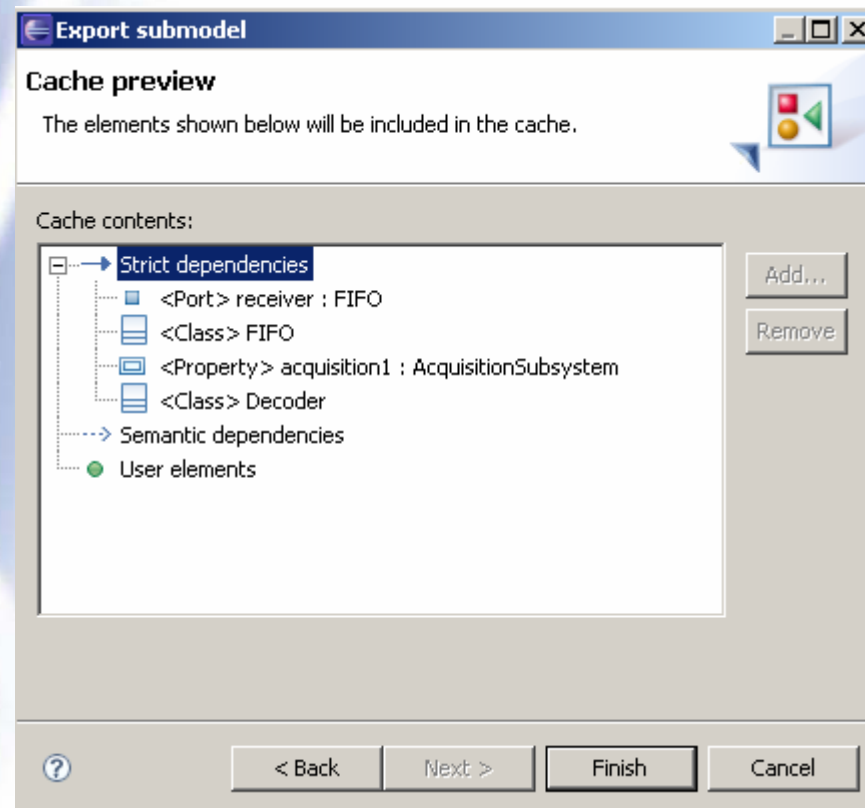
- Goal : give to a supplier a sub-model without parent model
  - ▶ → to be able to export the sub-model + its dependencies with parent model
- « Export » command provided for every sub-model
  - ▶ Note : requires usage of « control » command beforehand
  - ▶ Sub-model can be stored in a special place (export subdirectory for instance)





## Work in « out-sourcing » mode (2/8)

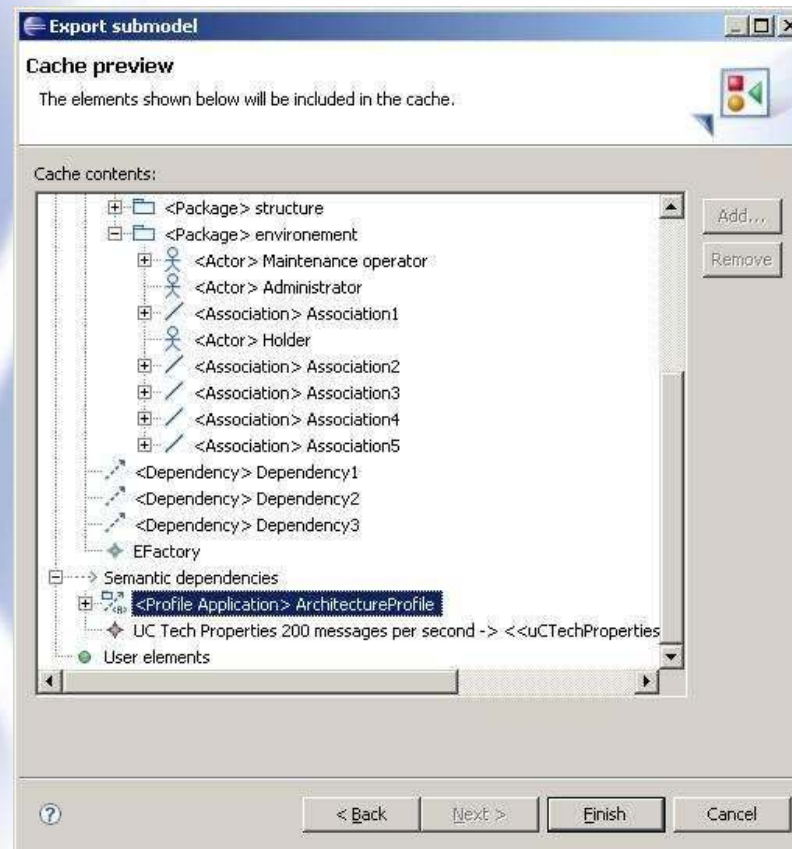
- A cache model is created with its dependencies to the parent model
  - ▶ In this example, “FIFO” and “Decoder” classes are added to the cache
  - ▶ It is also possible to manually add other items





## Work in « out-sourcing » mode (3/8)

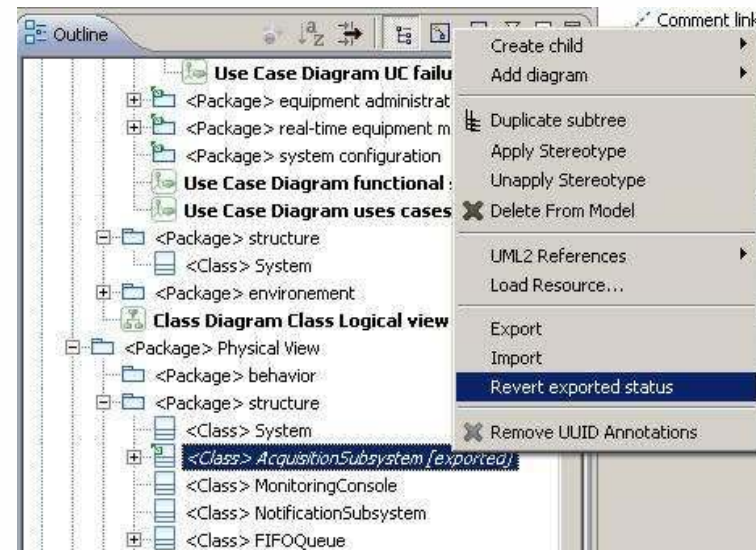
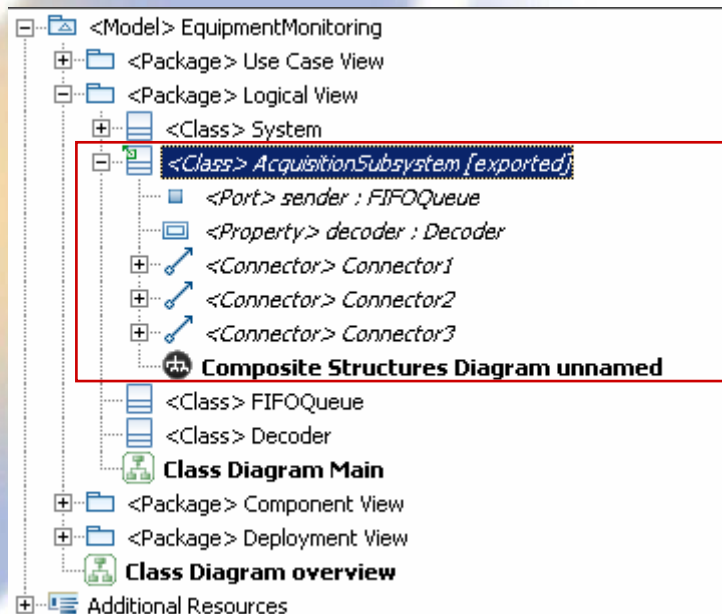
- Other example : export a functional domain
  - ▶ Here the cache contains three actors, associations with UC and one profile





## Work in « out-sourcing » mode (4/8)

- Finalization of the export is done by saving
  - ▶ The status of the sub model is stored in a properties file
- The exported items can no longer be modified in the parent model
  - ▶ Remark : in fact modifications are possible but cannot be saved

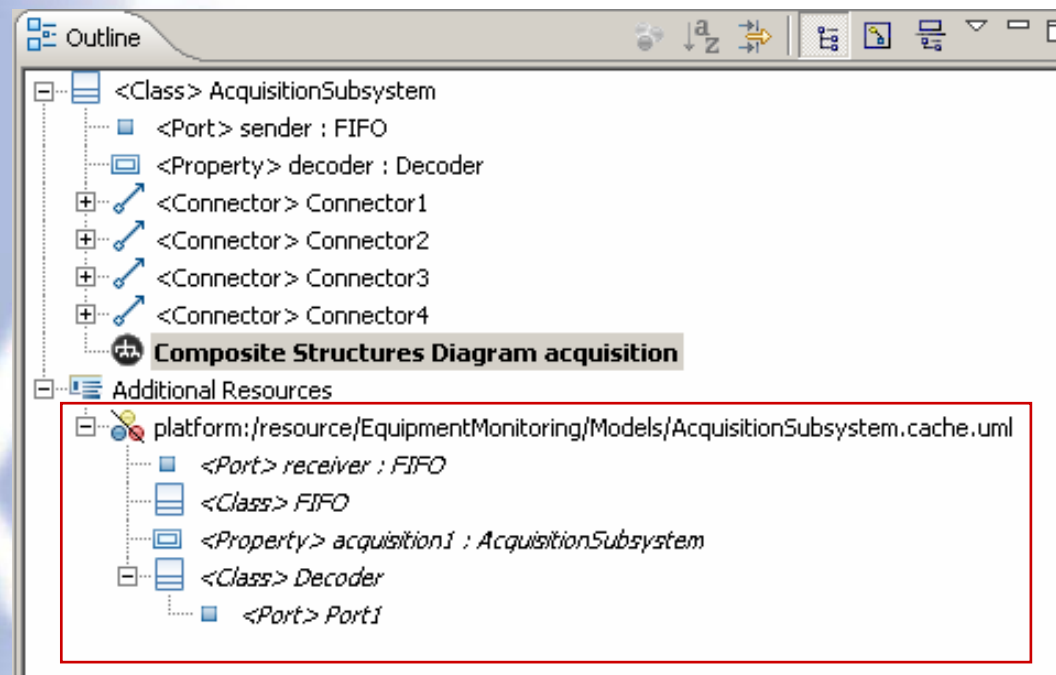


Remark : It is possible to cancel the export (go back)



## Work in « out-sourcing » mode (5/8)

- The exported sub-model becomes completely autonomous
  - ▶ the dependencies with the parent model are found in its cache

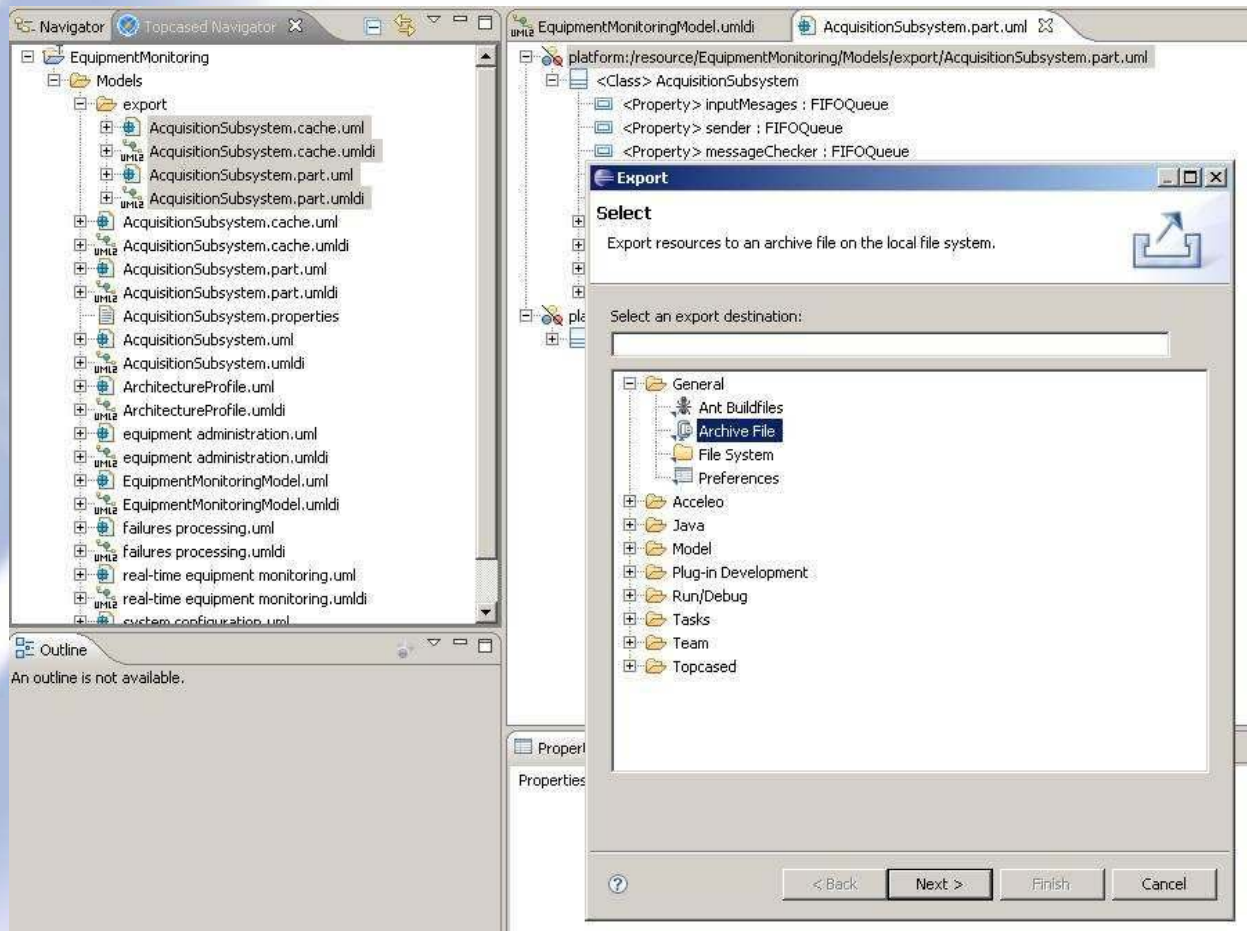






## Work in « out-sourcing » mode (6/8)

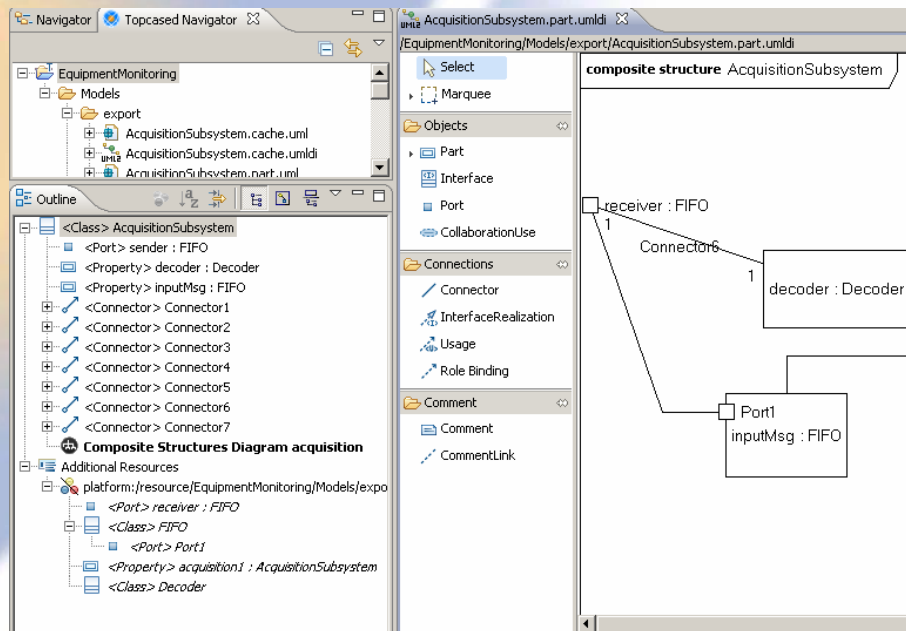
- The sub-model thus exported can be sent to the sub-contractor (with its cache)
  - ▶ Select different files, then Eclipse « export » command



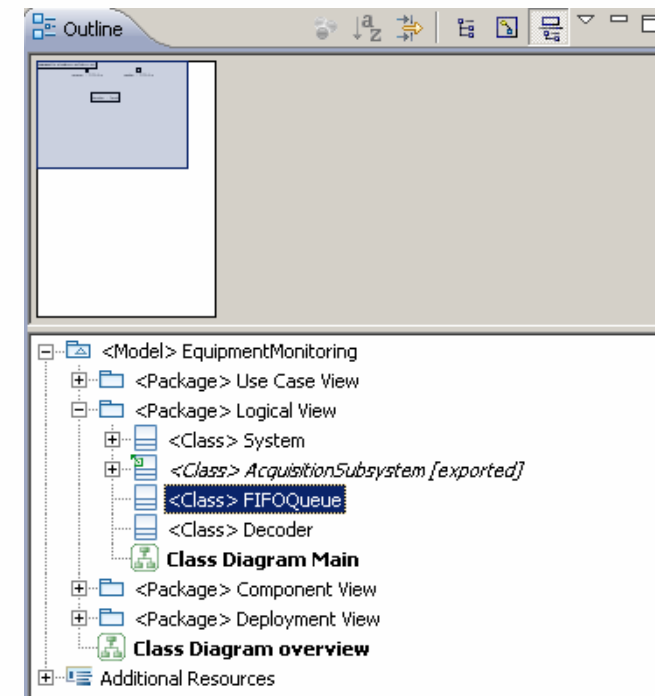


## Work in « out-sourcing » mode (7/8)

- Modifications can be done by the supplier in the sub-model
  - ▶ Example : addition of two parts of type “FIFO”
  - ▶ Modifications are then sent to the integrator
- During this period, the main model might have evolved
  - ▶ Example : FIFO has been renamed “FIFOQueue”



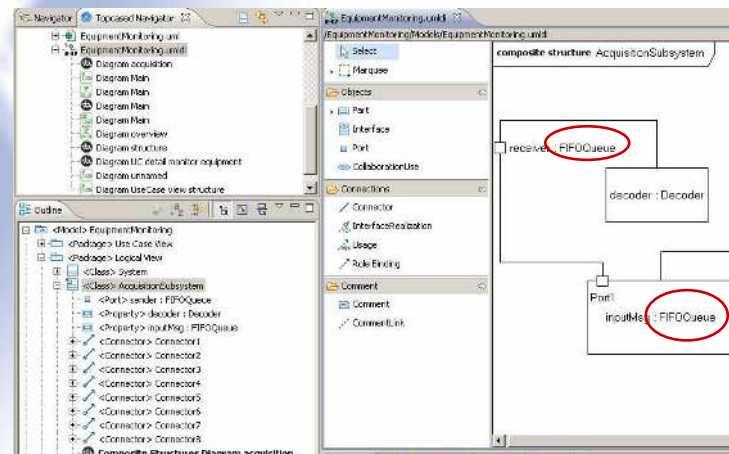
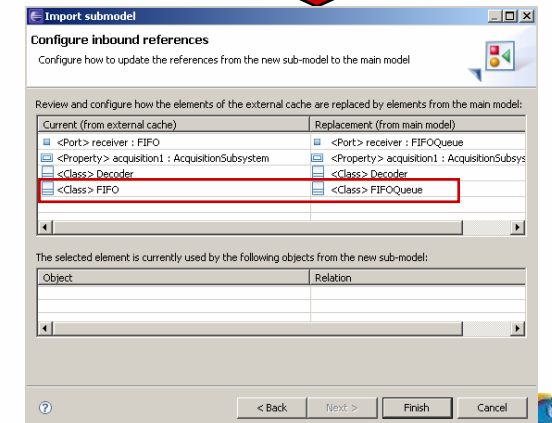
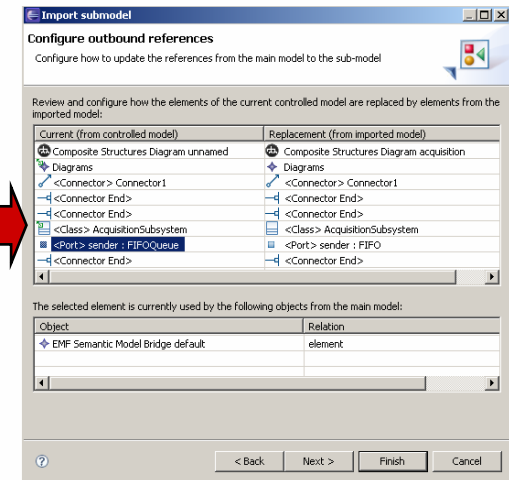
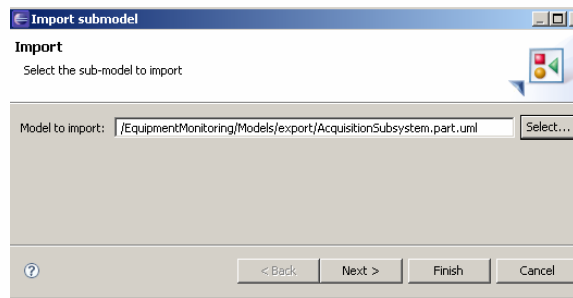
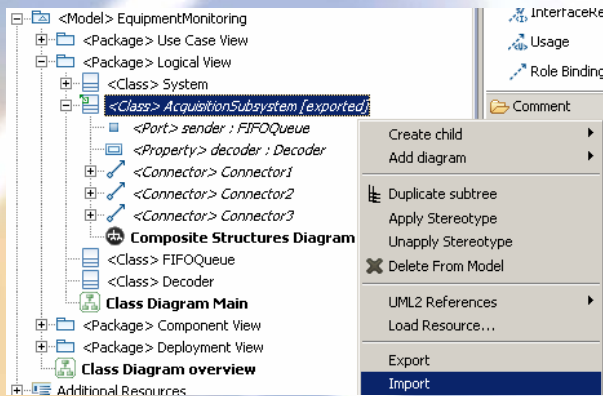
*In parallel*





# Work in « out-sourcing » mode (8/8)

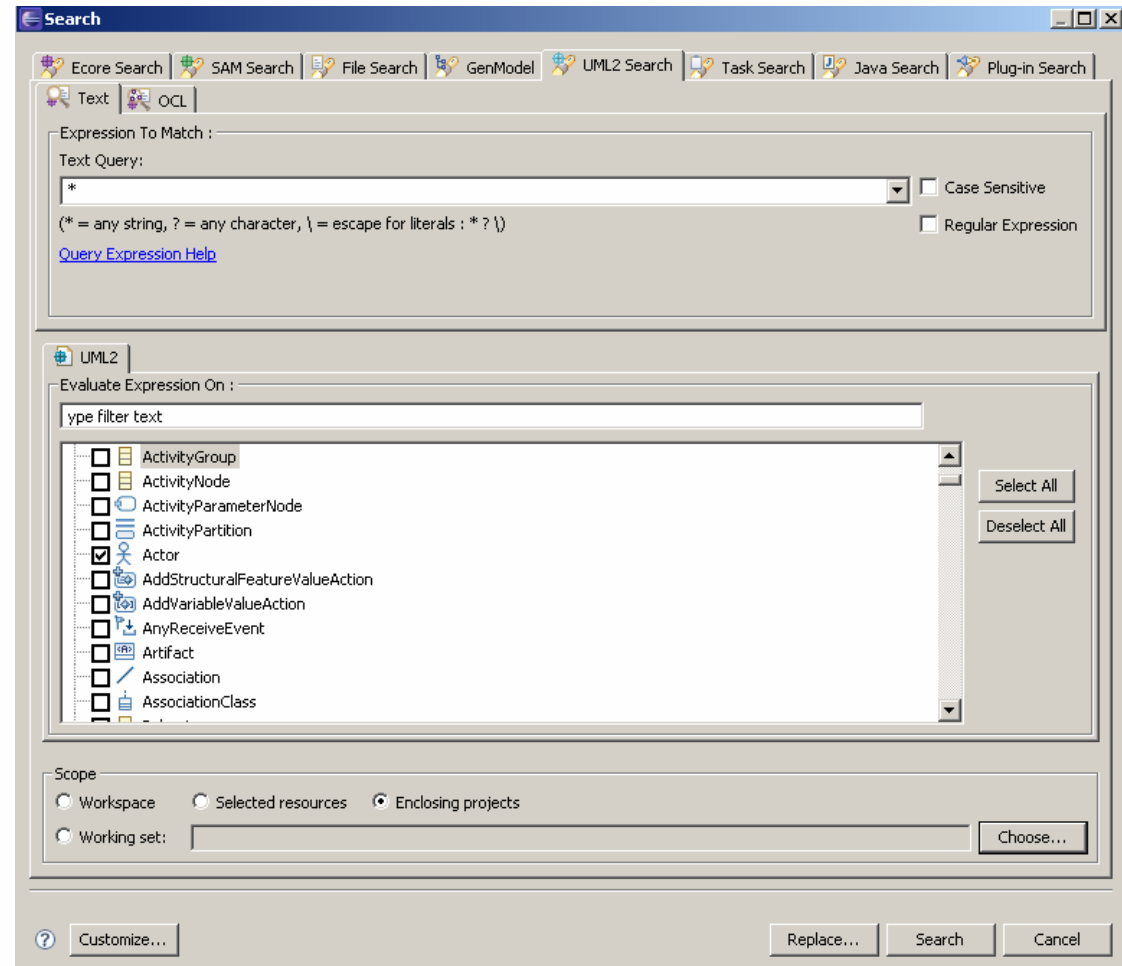
- « Import » command from the main model
  - ▶ Find the modifications in dependencies between export and import





## Search the models (1/2)

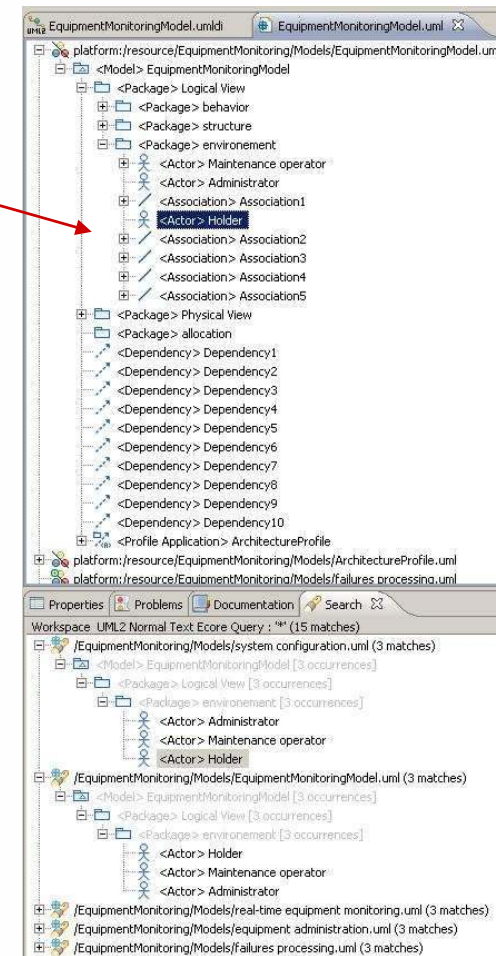
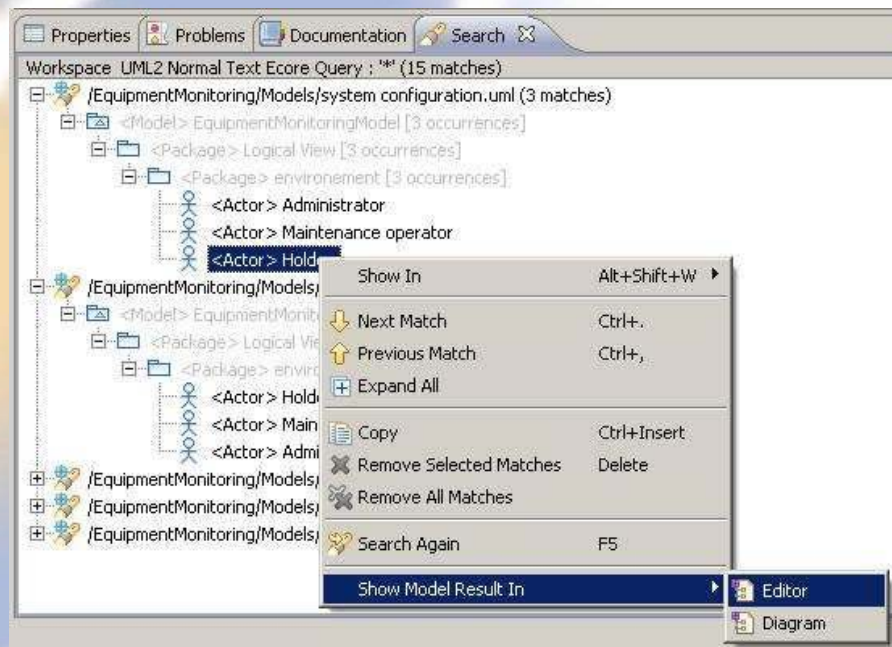
- Filter by language
- Text or Xpath mentioned
- Different scopes
  - ▶ Workspace
  - ▶ Working set
  - ▶ ...
- UML search example
  - ▶ Searching actors...
  - ... in some given projects





## Search the models (2/2)

- Results provided in the search view
- Double click → place on the UML model



- Placement on the diagrams not possible for 2.0