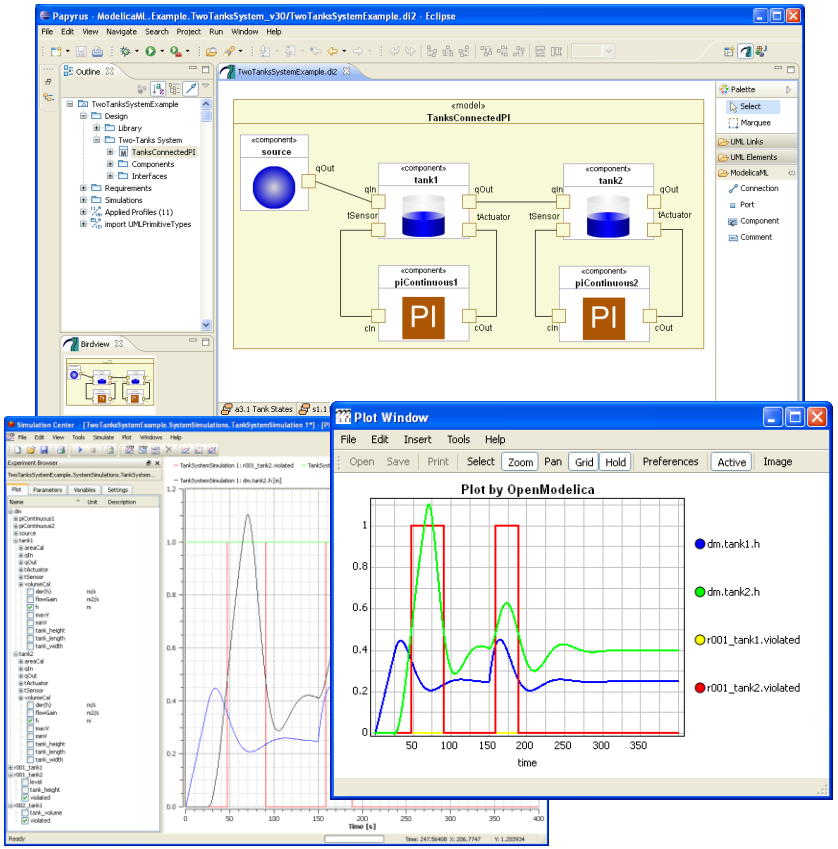
ModelicaML – Tutorial

Getting Started

Wladimir Schamai
EADS Innovation Works
Systems Engineering

ModelicaML: Technology

1 System Modeling with ModelicaML



2 Modelica Code Generation

```
within TwoTanks;

function limitValue

  input Real pMin;
  input Real pMax;
  output Real pLim;
  input Real p;

  algorithm
  // code generated from the Activity "algorithm (diagram)"
  // Activity.name: "algorithm (diagram)"
    if p < pMin then
      pLim := pMin; // OpaqueAction.name: "pLim := pMin;"
    elseif p > pMax then
```

```
within DesignModels;

model Tank

  TwoTanksSystemExample.DesignModels.ReadSignal tSensor; //Connector, sensor reading tank le
  TwoTanksSystemExample.DesignModels.ActSignal tActuator; //Connector, actuator controlling
  TwoTanksSystemExample.DesignModels.LiquidFlow qIn; //Connector, flow (m3/s) through input
  TwoTanksSystemExample.DesignModels.LiquidFlow qOut; //Connector, flow (m3/s) through outpu
  parameter Real flowGain(unit = "m2/s") = 0.05;
  parameter Real minV = 0; //Limits for output valve flow
  parameter Real maxV = 10; //Limits for output valve flow
  Real h(start = 0.0, unit = "m"); //Tank level
  parameter Real tank_height = 0.6;
  parameter Real tank_width = 1;
  parameter Real tank_length = 1.3;
```

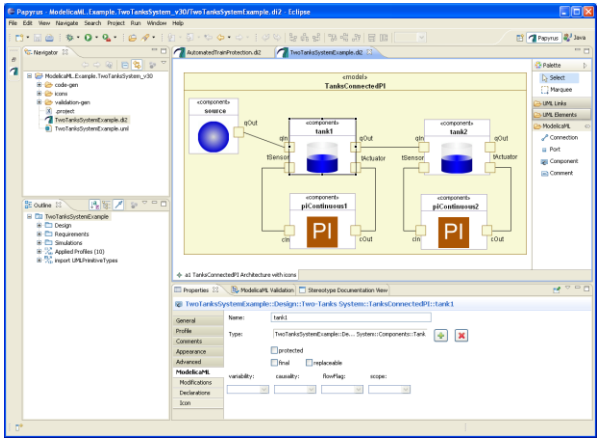
```
within DesignModels;

model TanksConnectedPI

  TwoTanksSystemExample.DesignModels.LiquidSource source(flowLevel = 0.02);
  TwoTanksSystemExample.DesignModels.Tank tank1;
  TwoTanksSystemExample.DesignModels.Tank tank2(tank_length = 1);
  TwoTanksSystemExample.DesignModels.PIcontinuousController piContinuous1(ref = 0.25);
  TwoTanksSystemExample.DesignModels.PIcontinuousController piContinuous2(ref = 0.4);

  equation
    connect(source.qOut, tank1.qIn);
    connect(piContinuous1.cIn, tank1.tSensor);
    connect(piContinuous1.cOut, tank1.tActuator);
    connect(tank1.qOut, tank2.qIn);
    connect(piContinuous2.cIn, tank2.tSensor);
    connect(piContinuous2.cOut, tank2.tActuator);

end TanksConnectedPI;
```

3 System Simulation with Modelica Tools

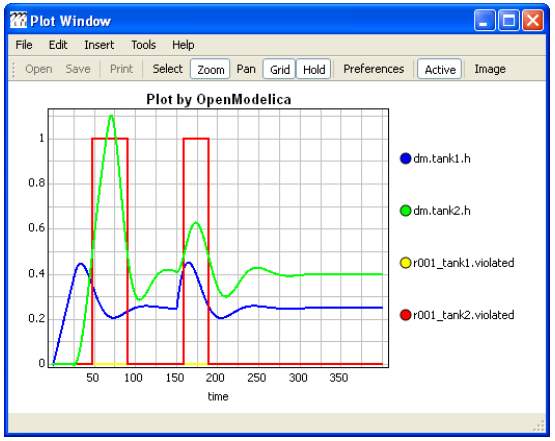
ModelicaML: Technology

Papyrus UML

ModelicaML Profile (Eclipse Plug-In)



Any Modelica Simulation Tool



Model to Text Transformation

Acceleo

ModelicaML Code Generator
(Eclipse Plug-In)

Generated Modelica Code
(.mo files)

EADS

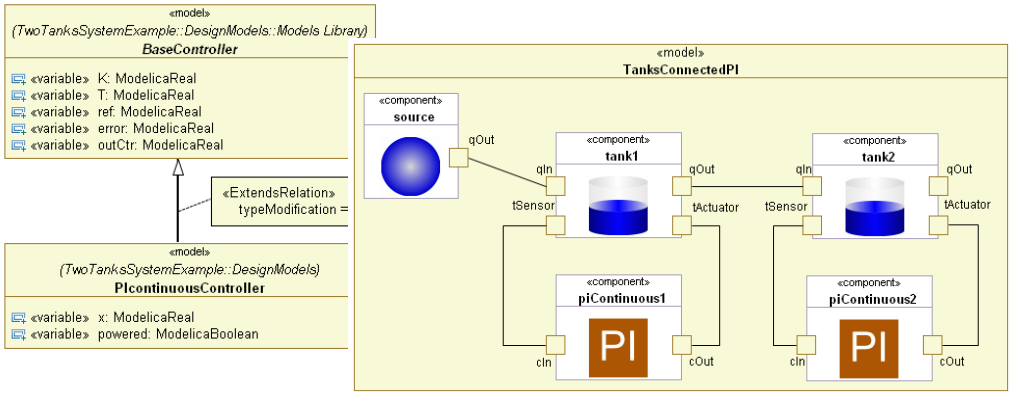
ModelicaML

General Description of the

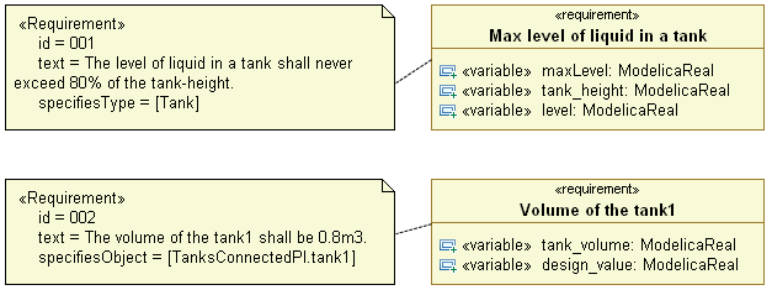
UML-Based Graphical Notation

ModelicaML: Graphical Notation

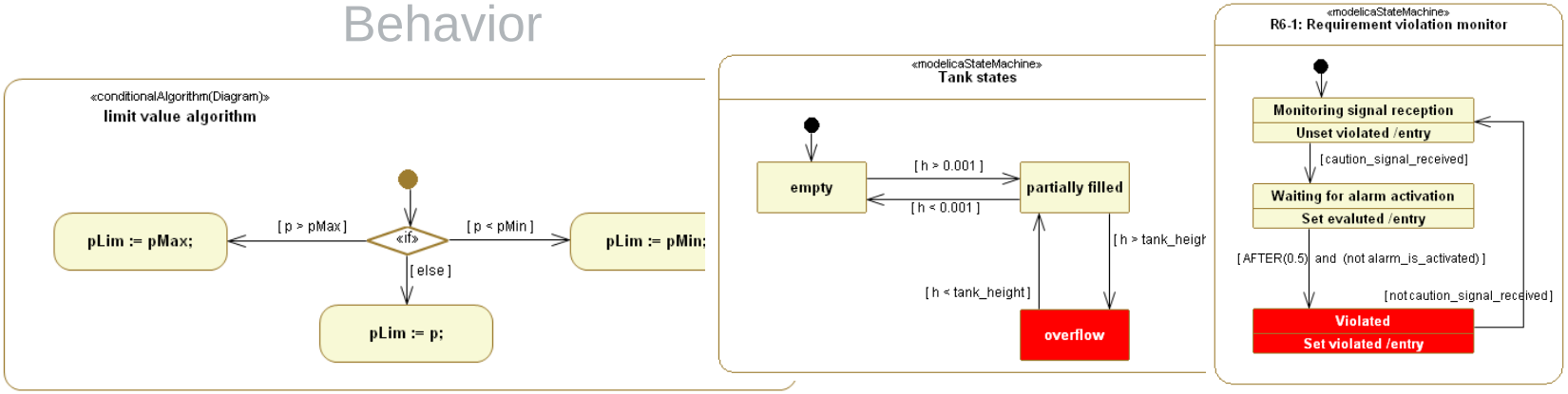
Structure



Requirements



Behavior



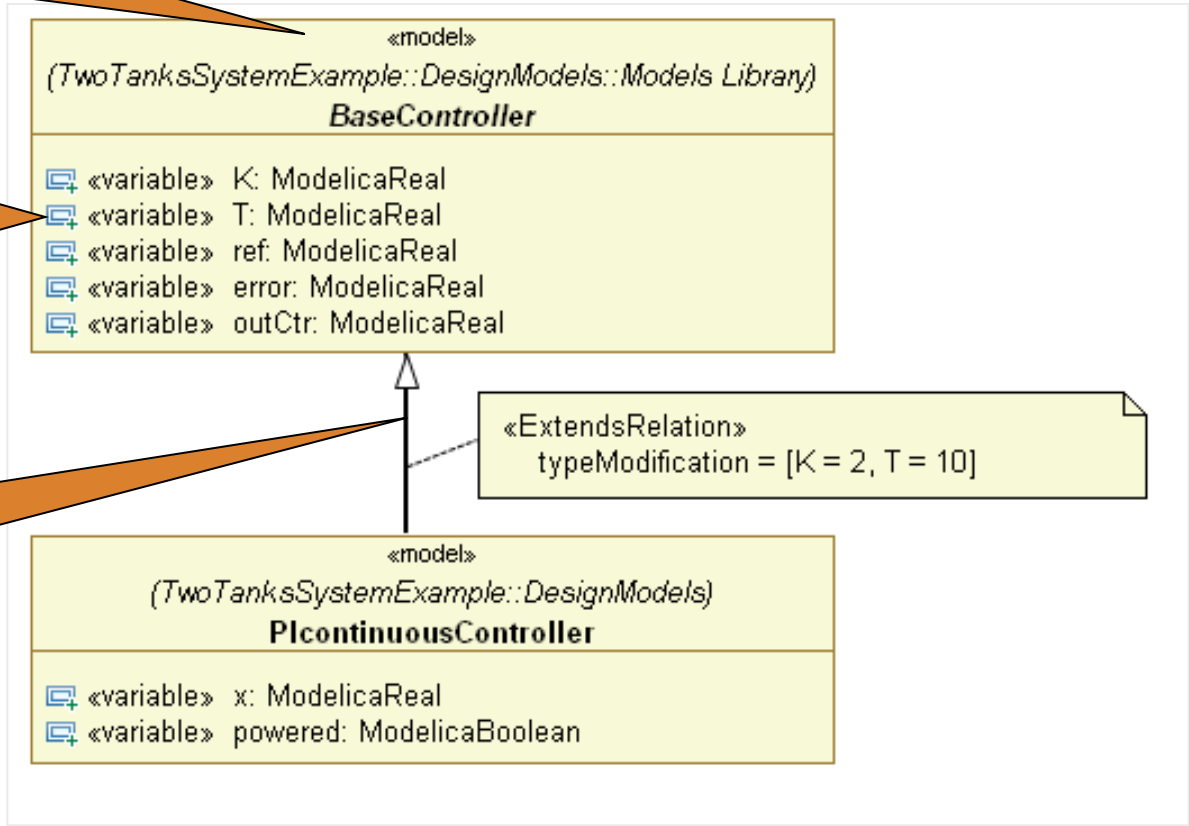
EADS

ModelicaML: Class Diagram

Class (model, block, record, connector)

variables (of primitive type) or **components** (of composite type)

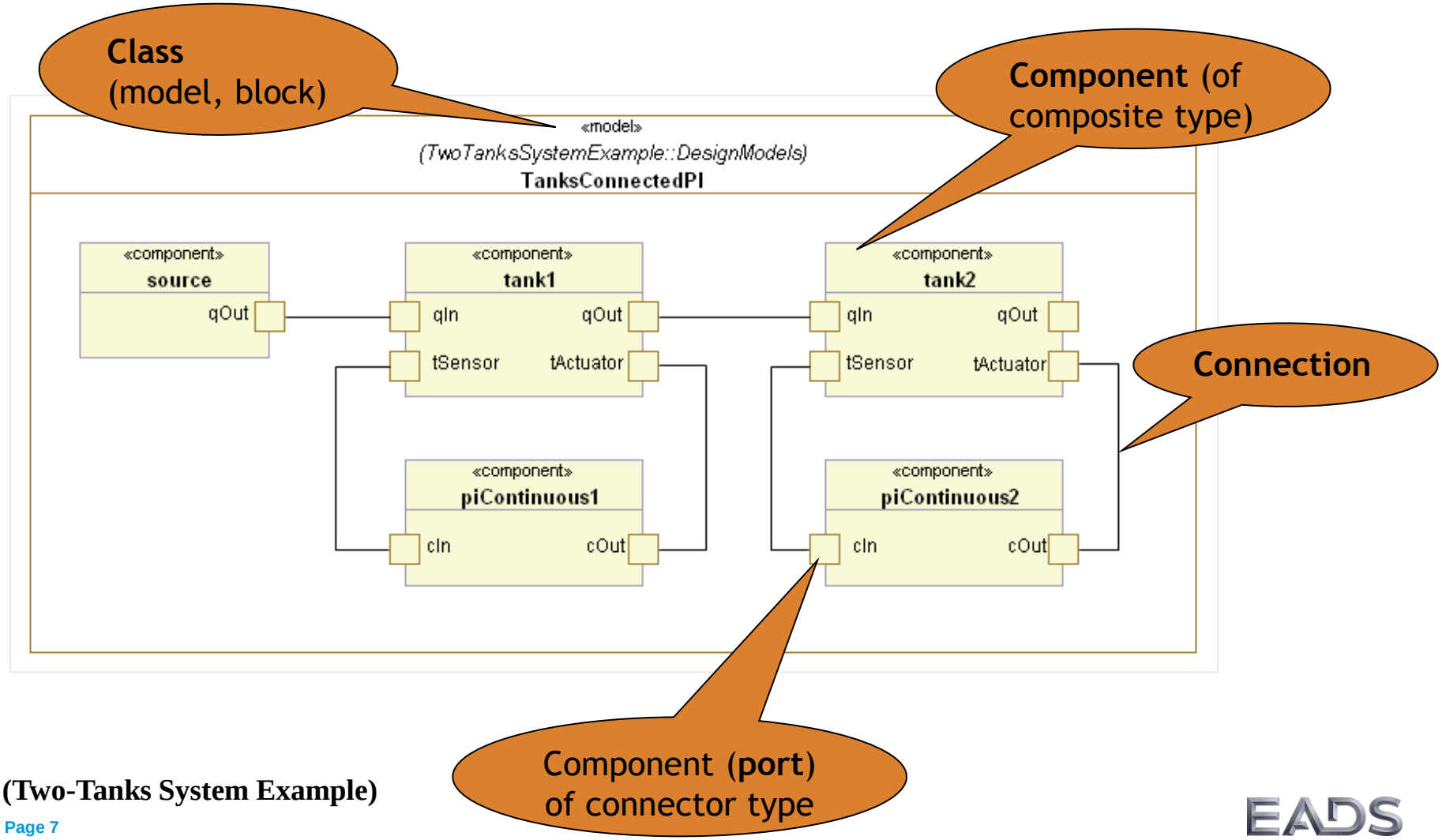
extends relation (with type modifications)



(Two-Tanks System Example)

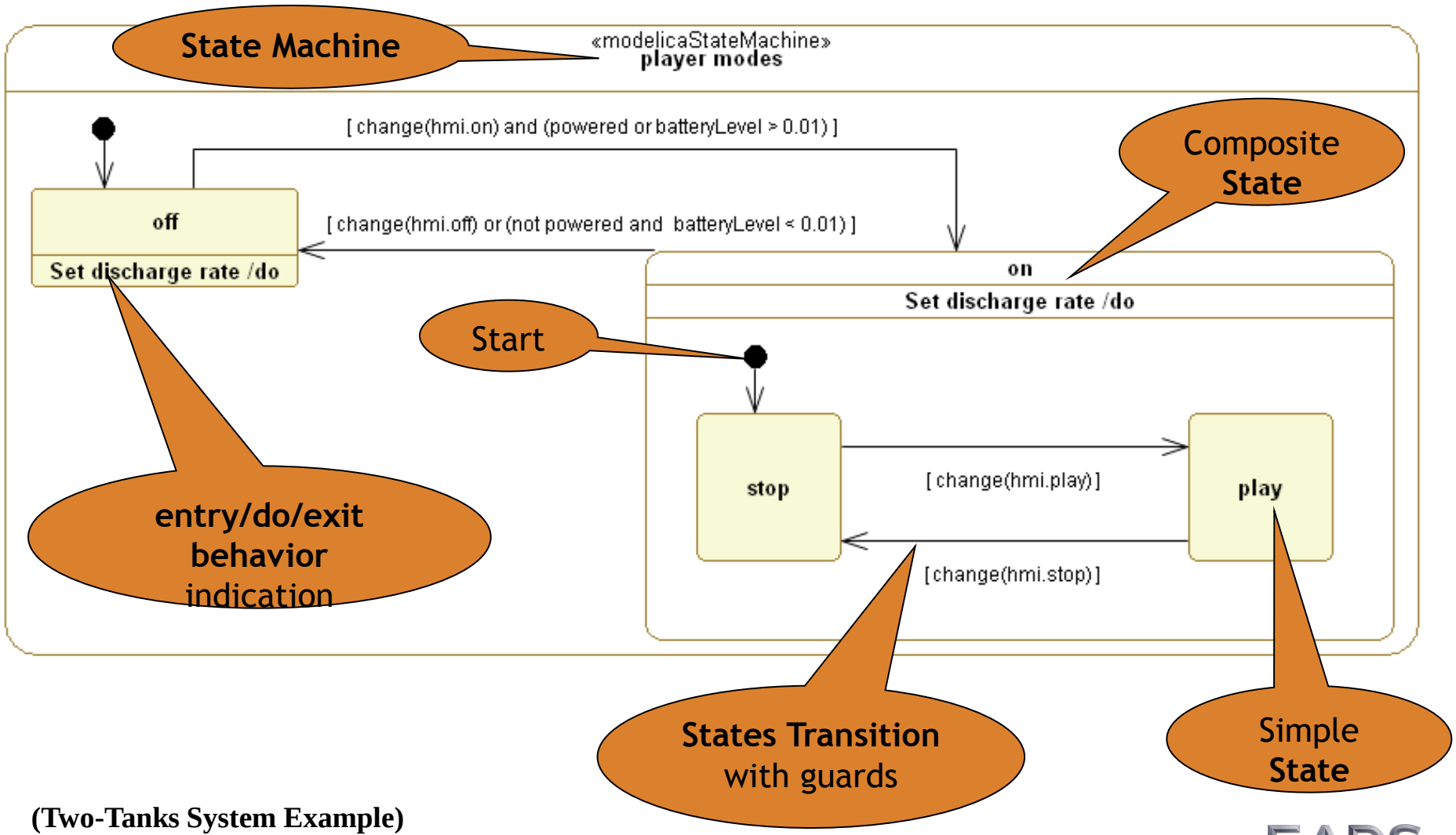
EADS

ModelicaML: Connection Diagram



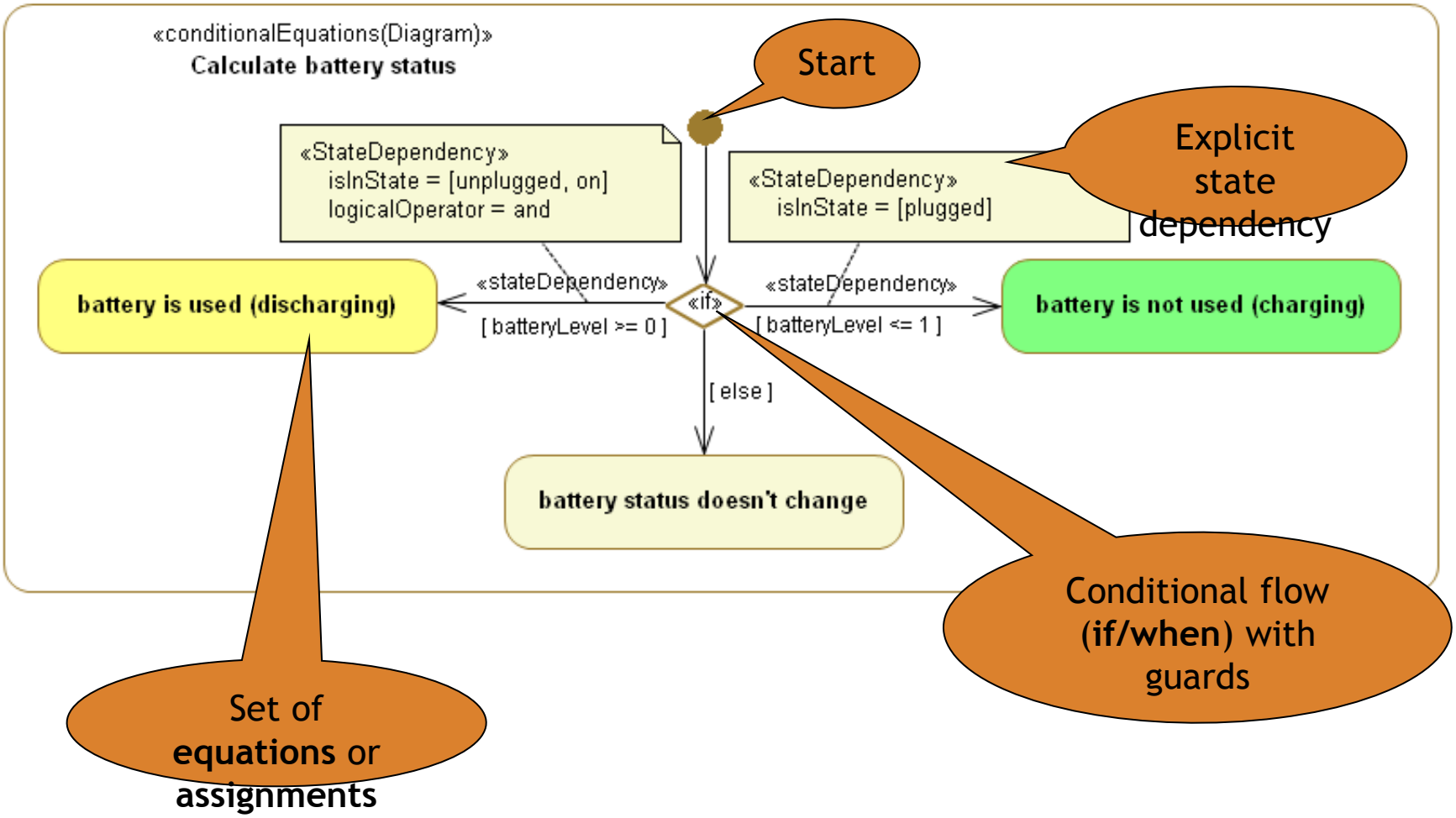
(Two-Tanks System Example)

ModelicaML: State Machine Diagram



(Two-Tanks System Example)

ModelicaML: Conditional Eq./Alg. Diagram



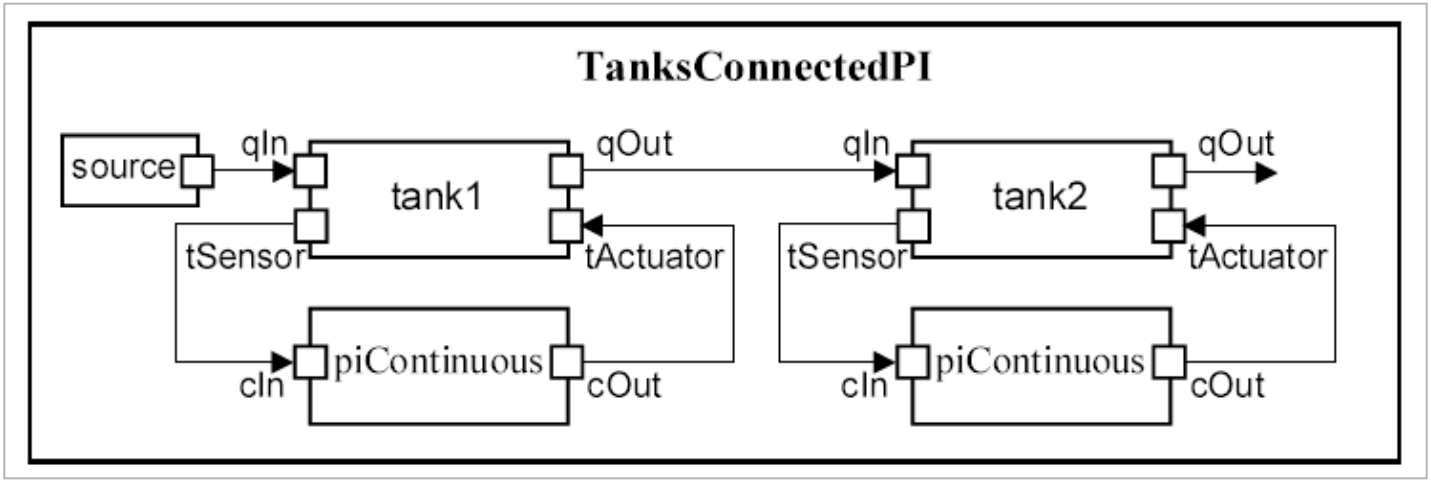
(Player Example)

ModelicaML

Hands-on Modeling Tutorial

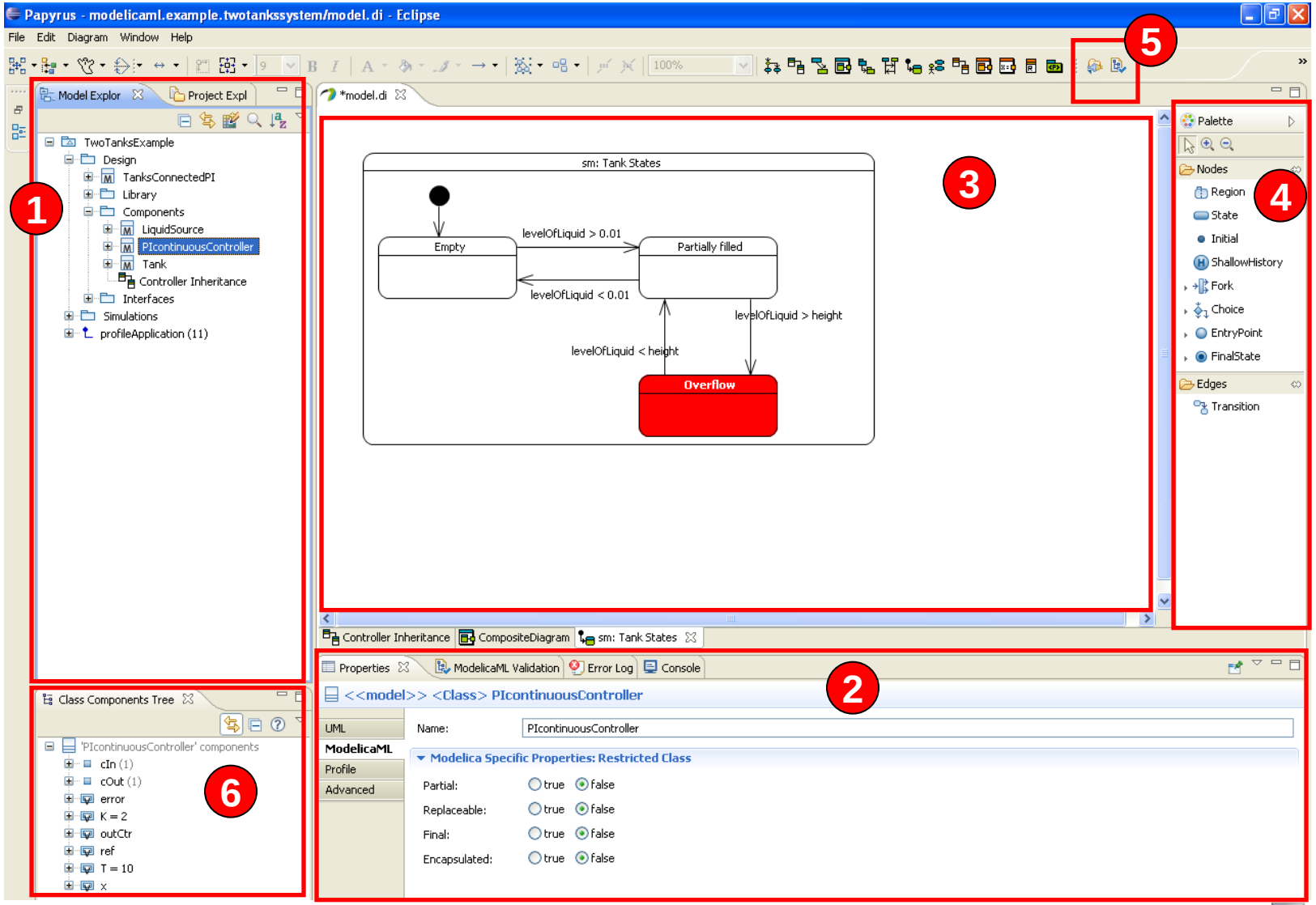
Example: Two Tanks System

- From “source” liquid flows into the “tank1”
- Controller “piContinuous1” controls the level of liquid in “tank1”, based on a predefined reference value, by opening and closing the tank outflow valve.
- Liquid flows from “tank1” into “tank2”
- Controller “piContinuous2” controls the level of liquid in “tank2”



Source: Peter Fritzson. Principles of Object-Oriented Modeling and Simulation with Modelica 2.1. Wiley-IEEE Press, 2004. page 391.

ModelicaML Papyrus MDT GUI Overview



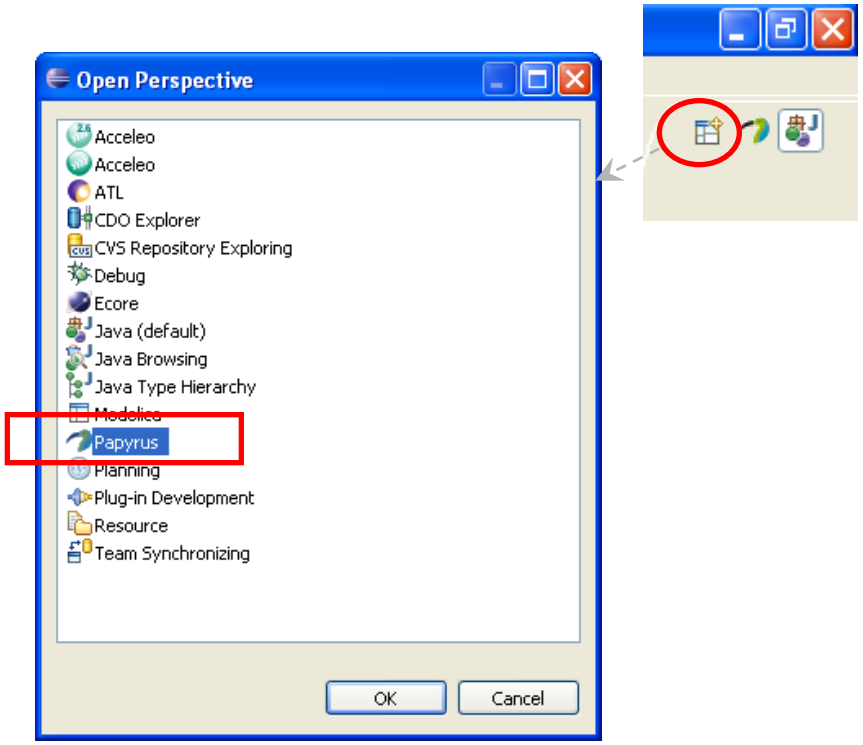
ModelicaML Papyrus MDT GUI Overview

1. Model Browser: Shows model elements
2. Properties View: Shows the properties of selected element
3. Diagram Editors (different UML-based diagrams)
4. Palette (different for each diagram)
5. ModelicaML code generation and validation buttons
6. Component tree: Shows the components hierarchy of the selected class

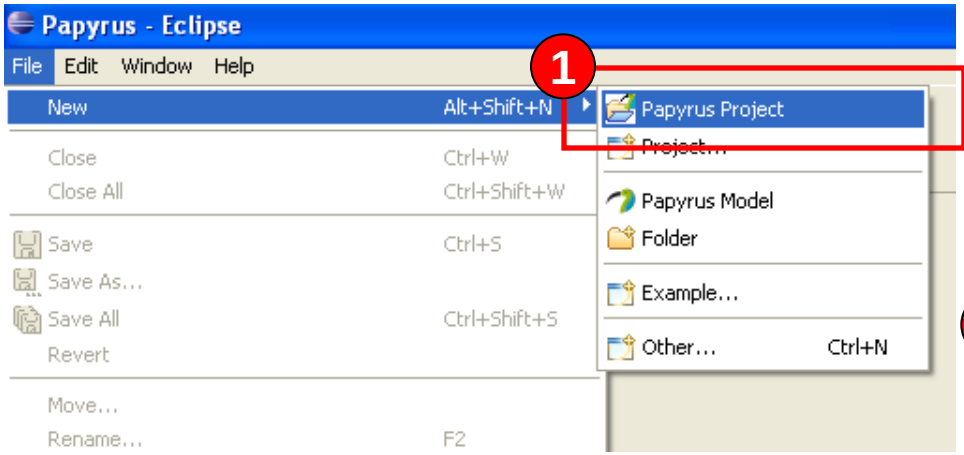
ModelicaML Project Setup

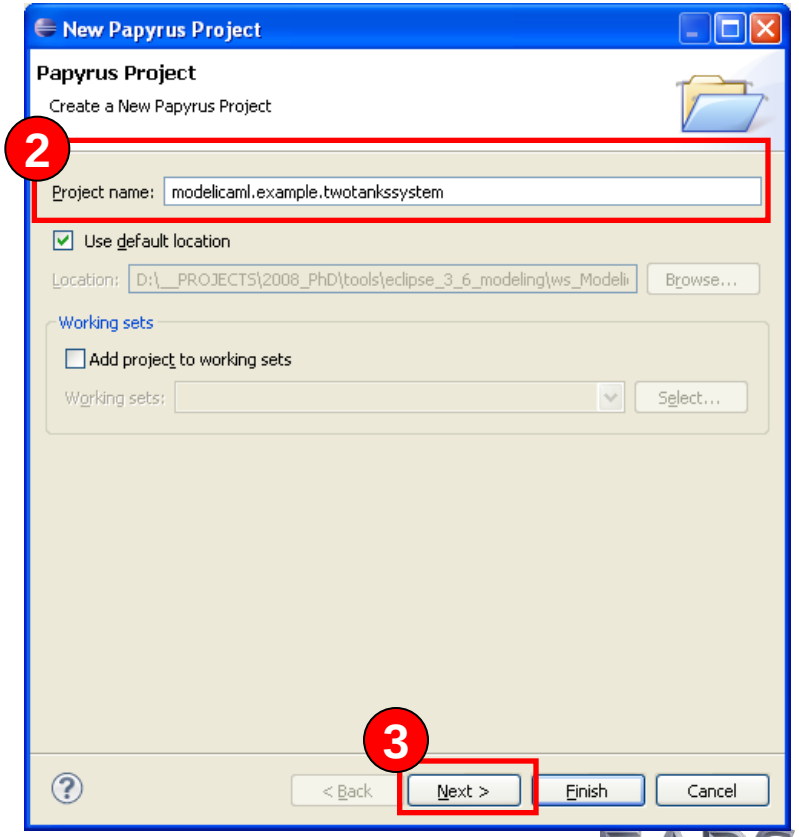
Create ModelicaML Project

- Open Eclipse
- Change the Perspective to Papyrus Perspective



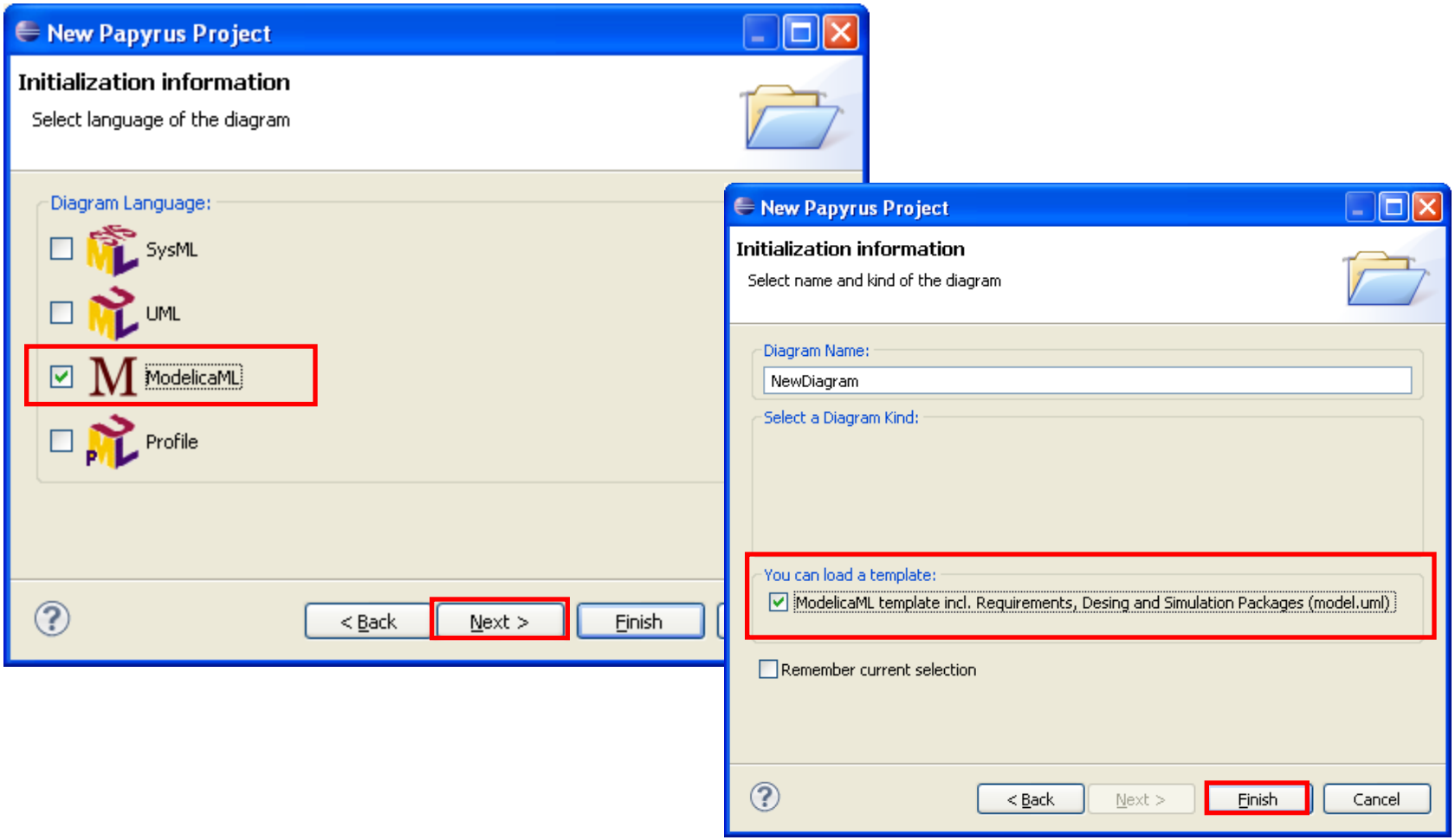
Create ModelicaML Project





Go to File -> Create …

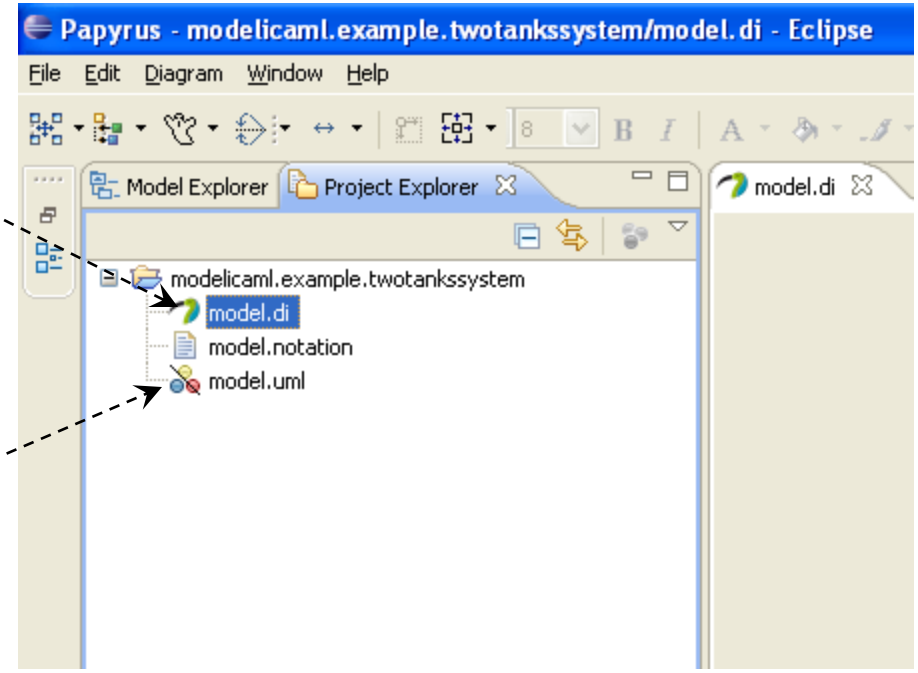
Create ModelicaML Project



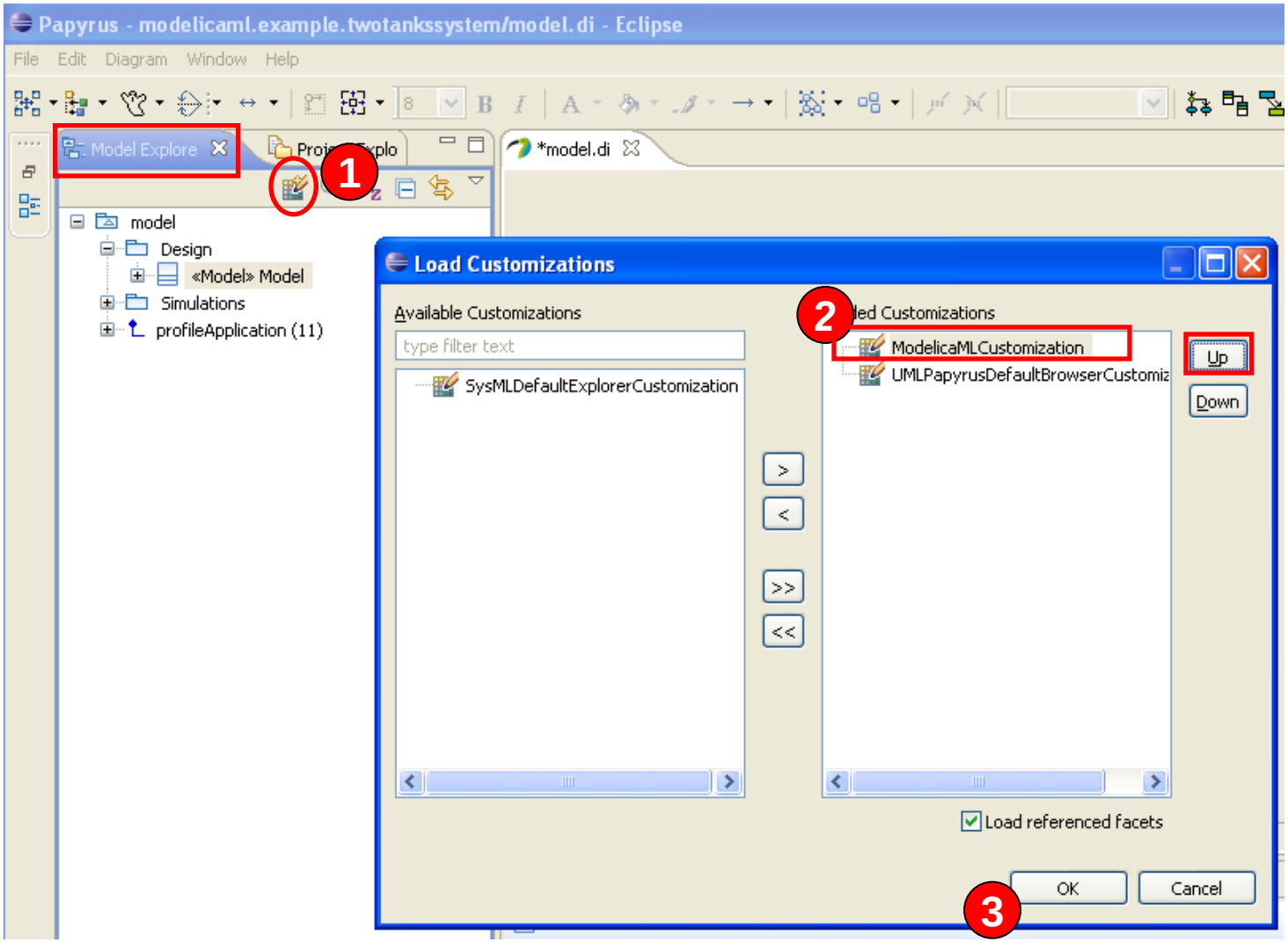
Papyrus Project Files

Diagram file
(can only be edited using Papyrus MDT)

Model file
(can be edited using any UML2 tool)

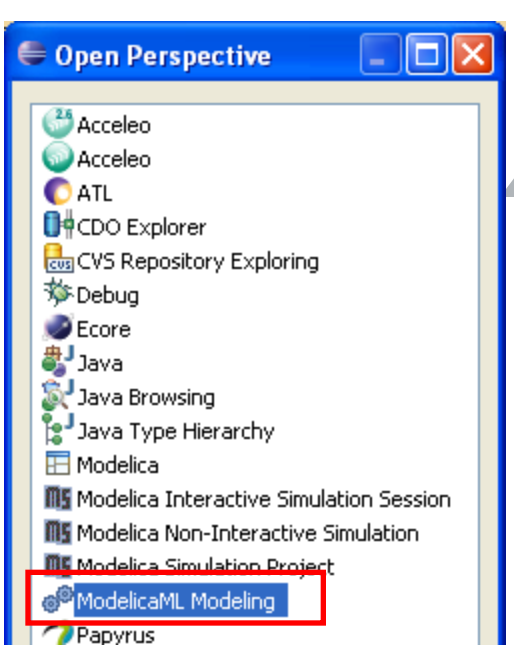


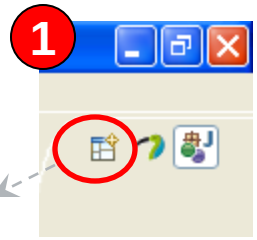
Configure Model Explorer



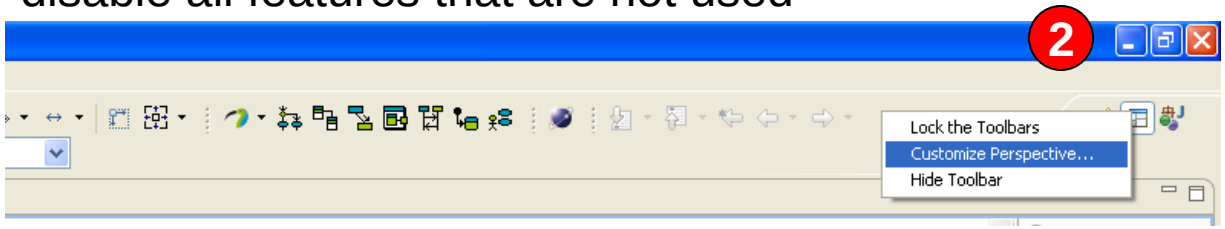
EADS

Change and Customize the Perspective

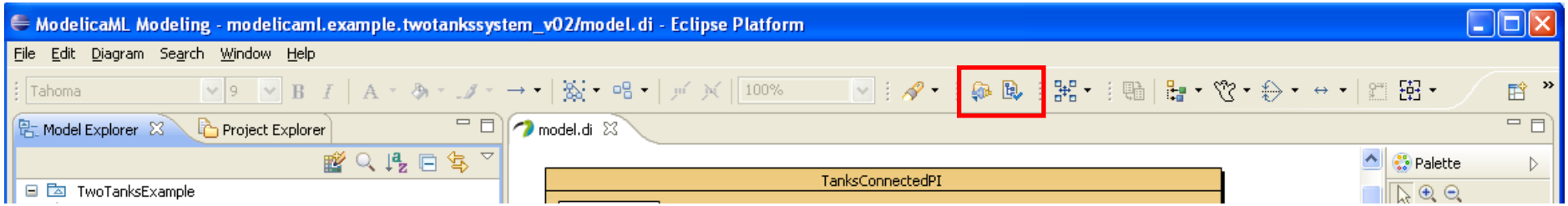




Select and customize the ModelicaML perspective – disable all features that are not used



Typical ModelicaML customized perspective:

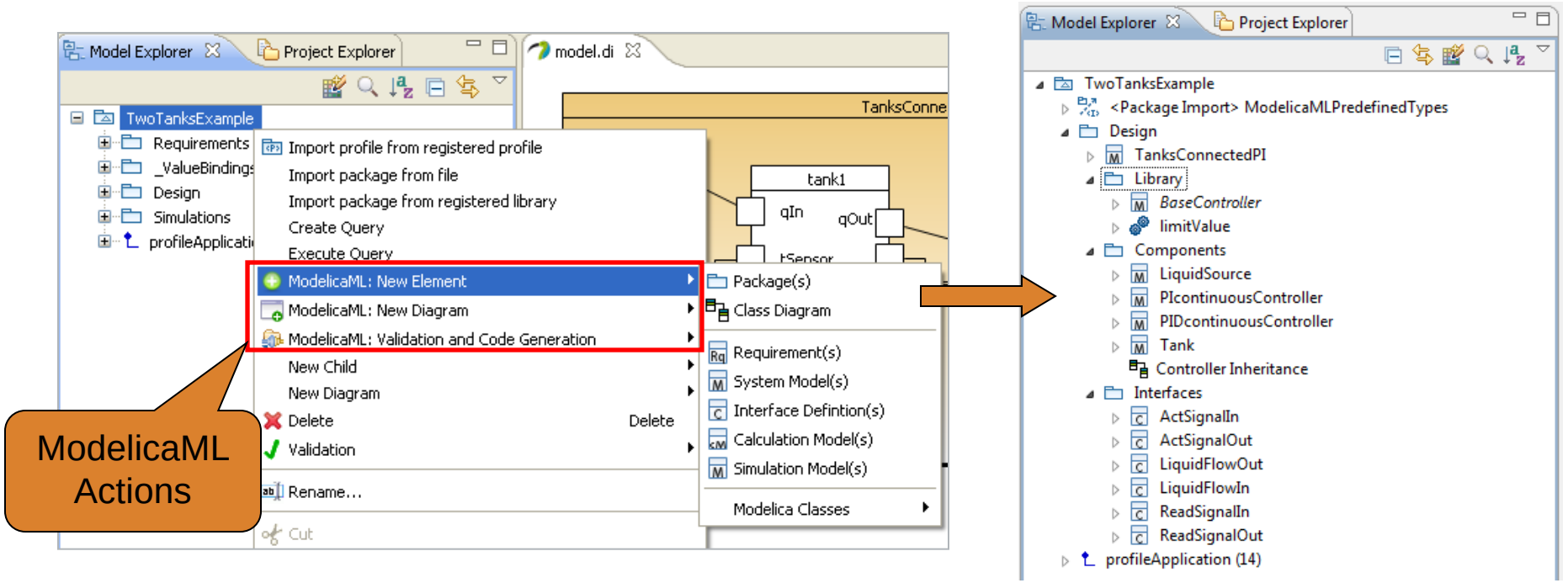


EADS

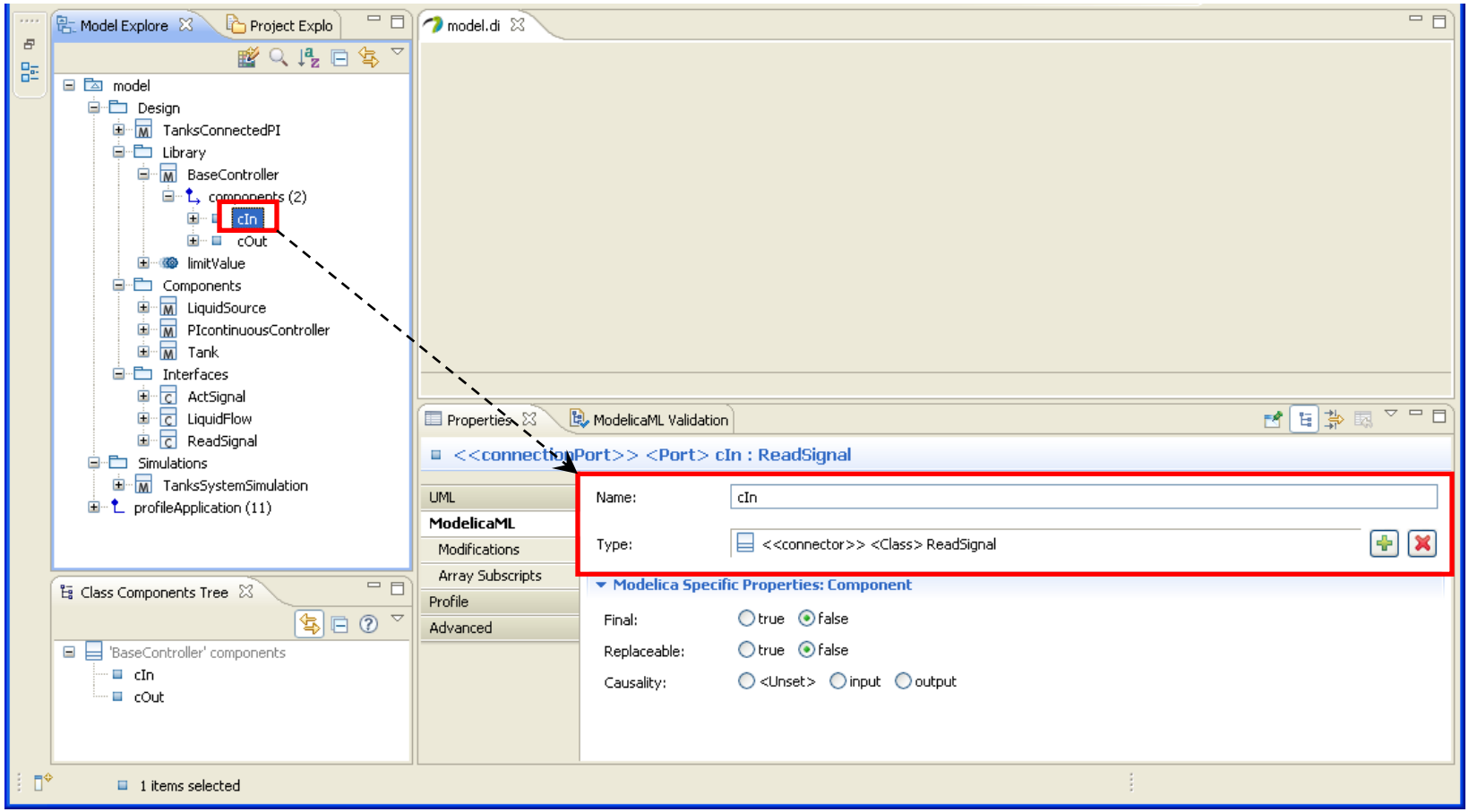
Model Setup

Create Model Structure

Create Packages and Classes using ModelicaML menus



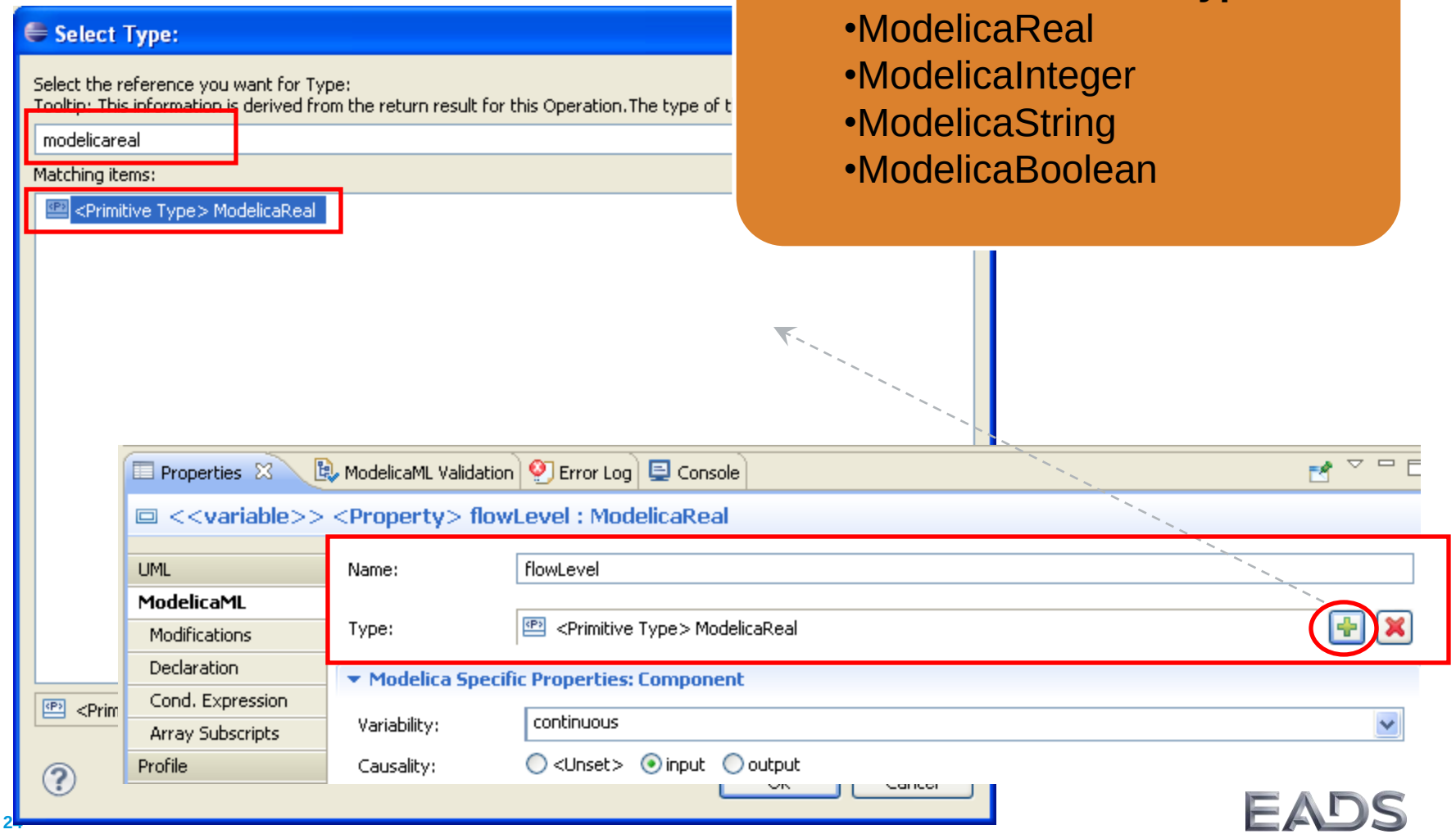
Hint: Setting type of components



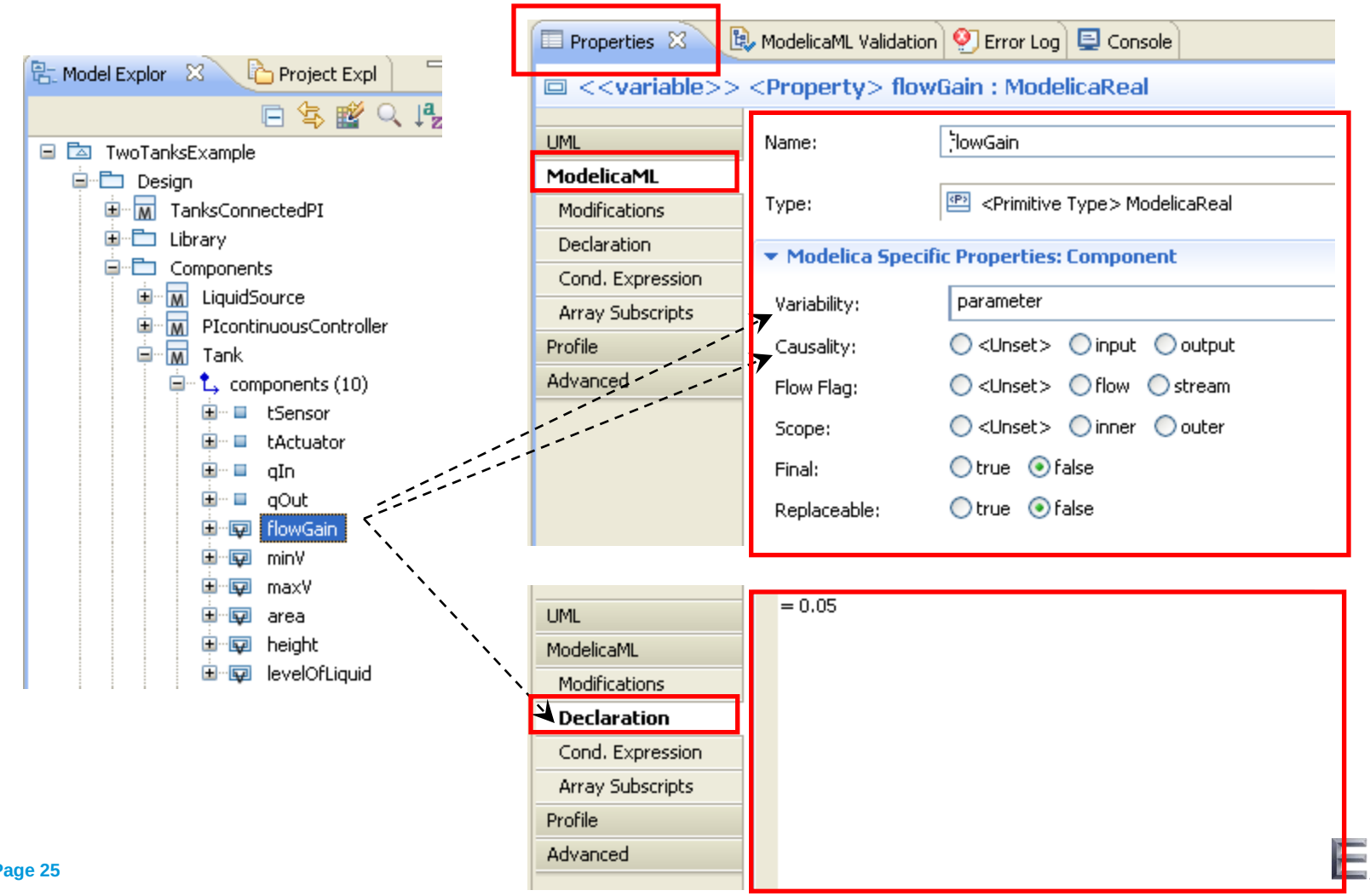
Hint: Setting type of components

ModelicaML Primitive Types are:

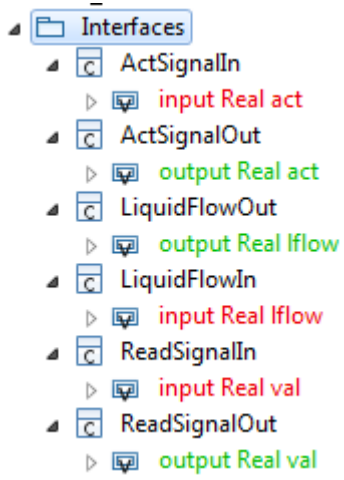
- ModelicaReal
- ModelicaInteger
- ModelicaString
- ModelicaBoolean

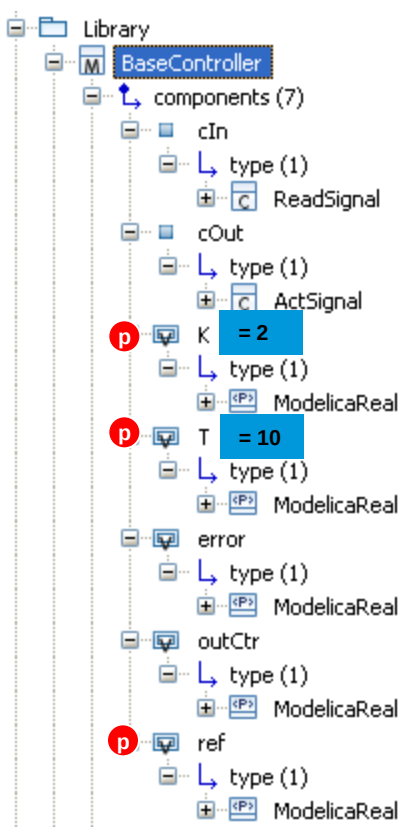


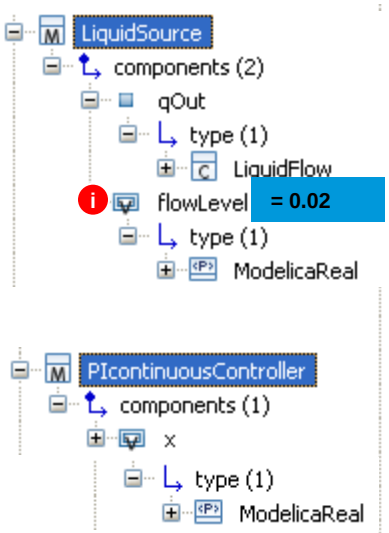
Hint: Setting of the component properties (Declaration, Causality, Variability, etc.)

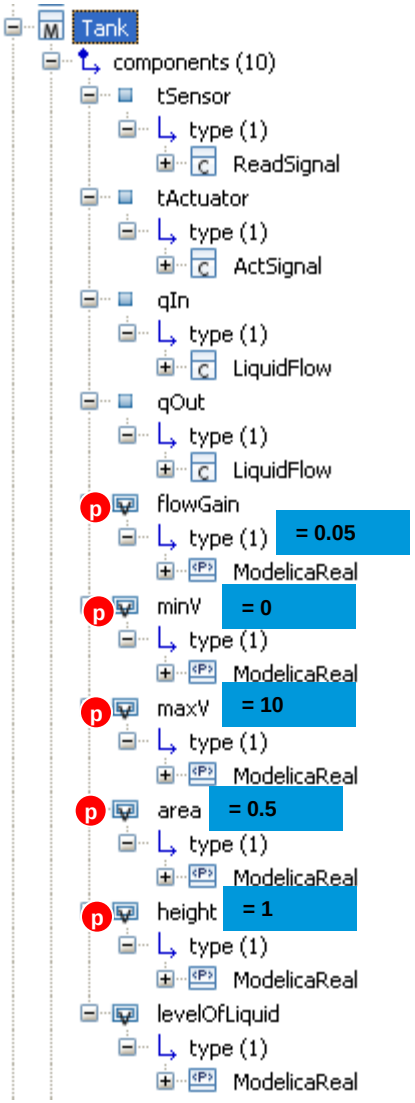


Create Class Components







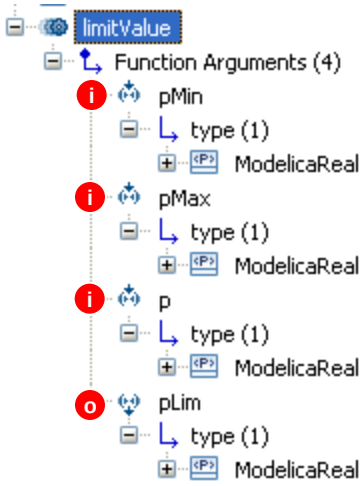


p => Set variability to “parameter”

i => Set causality to “input”

= 2 => Set declaration

Create Function Arguments

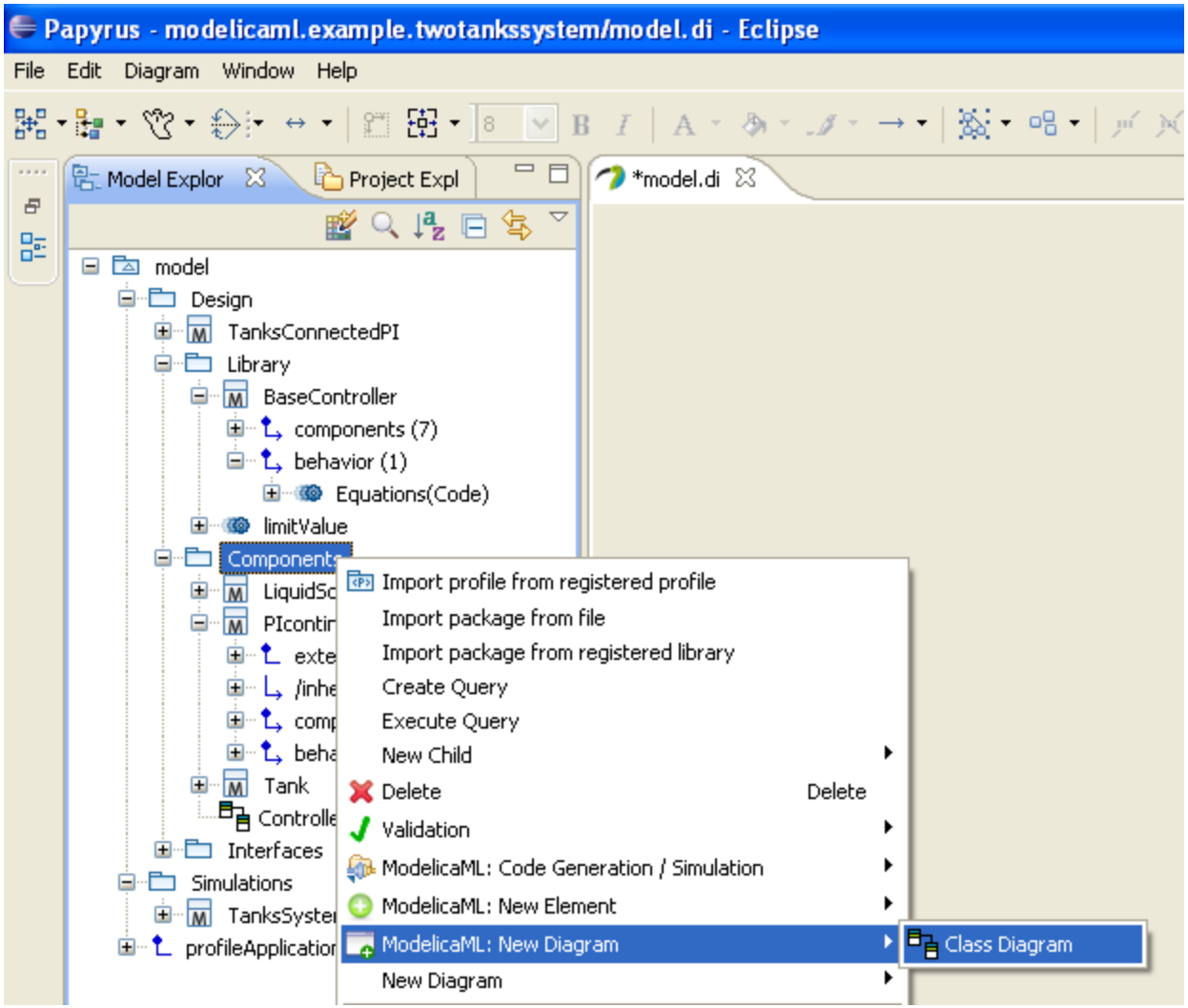


i => Set causality to “input”

o => Set causality to “output”

Inheritance/Extension Modeling

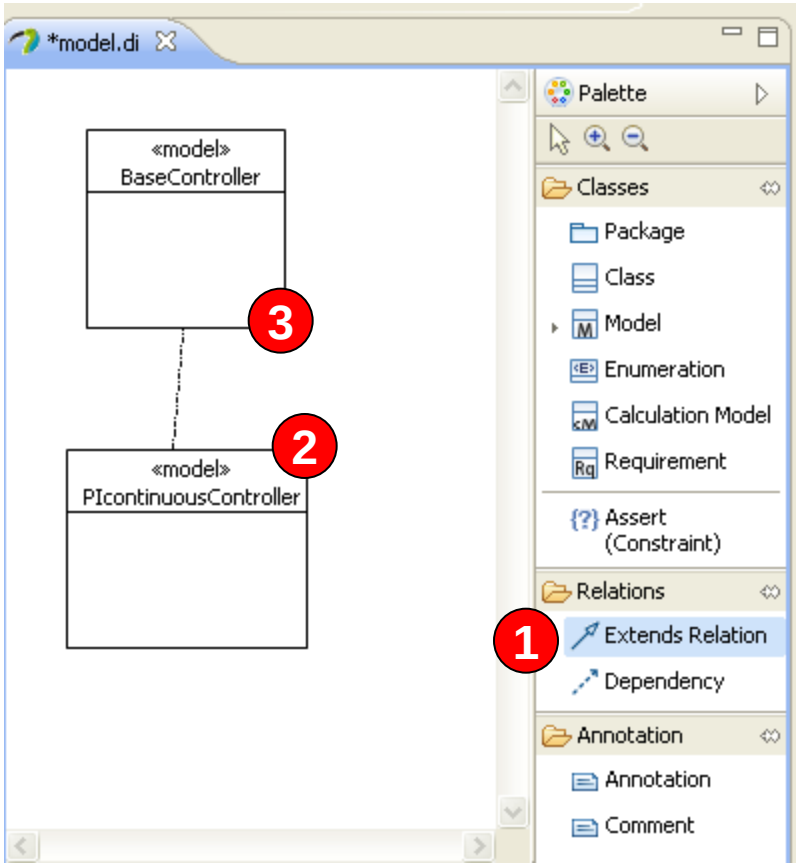
Create Class Diagram



General: Working with diagrams

Creating edges:

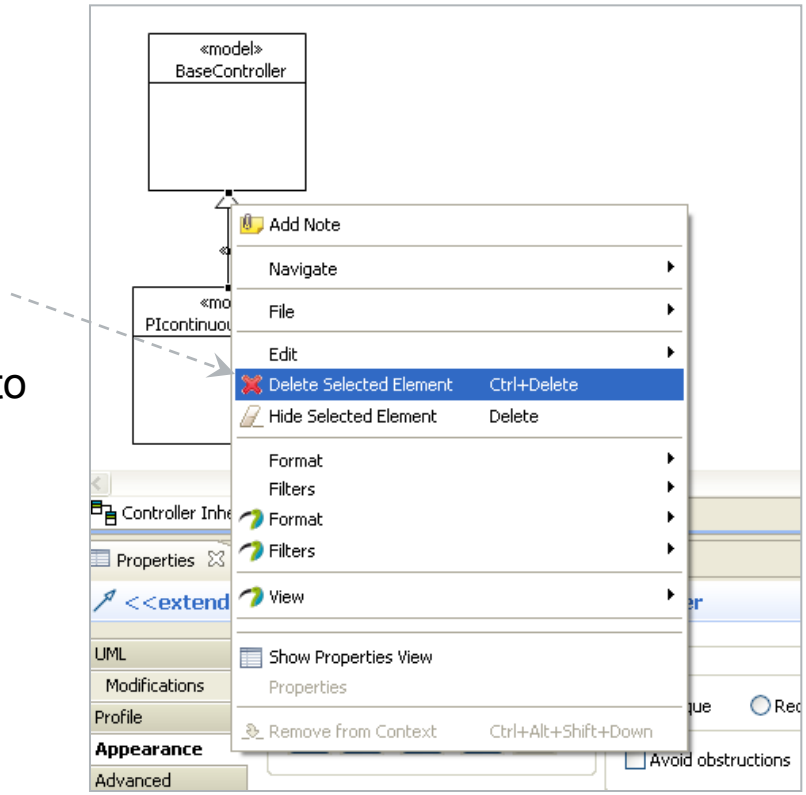
1. Select the palette tool
2. Click on the edge source element and hold the mouse button
3. Move the mouse to the target element and release the mouse button



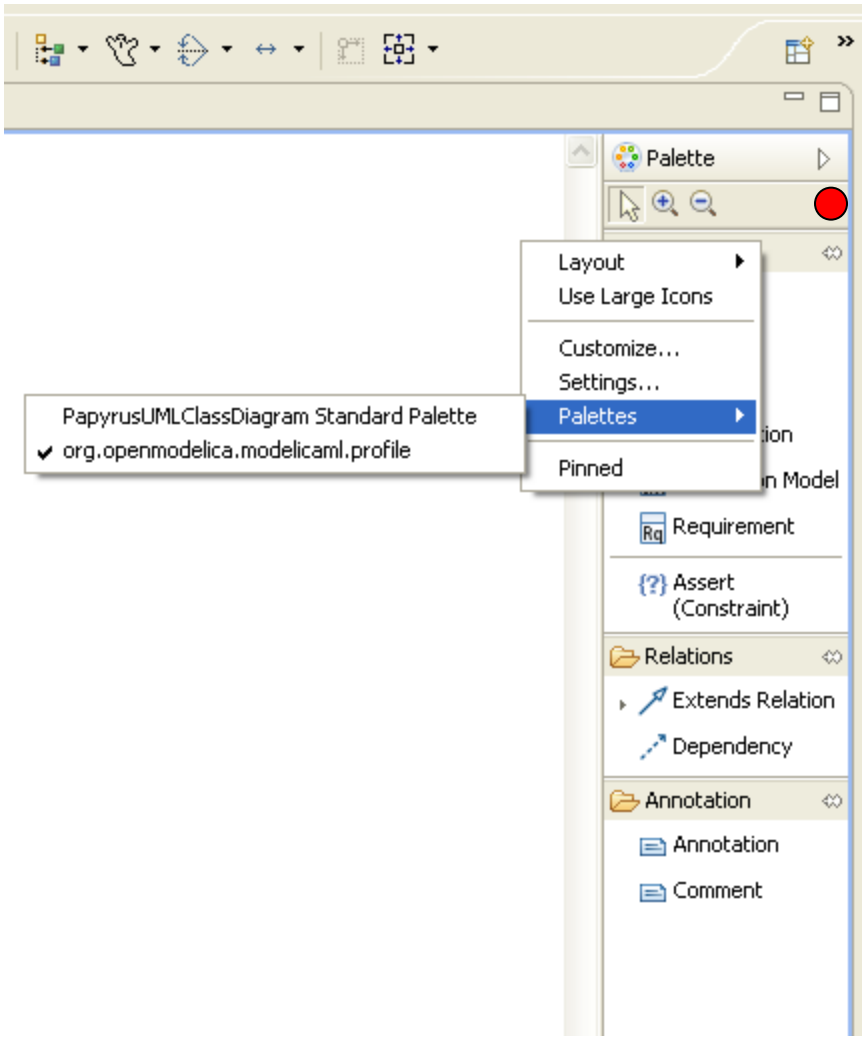
General: Working with diagrams

Deletion of elements:

- Right click on a diagram element
- (preferred) Select the option “Delete Selected Element” in order to delete it from the model. This is recommended in order to keep the model and the diagram consistent
- Select the option “Hide Selected Element” in order to remove the element from the diagram. The element will still exist in the model and can be shown on the diagram by drag&drop.



Configure Diagram Palette

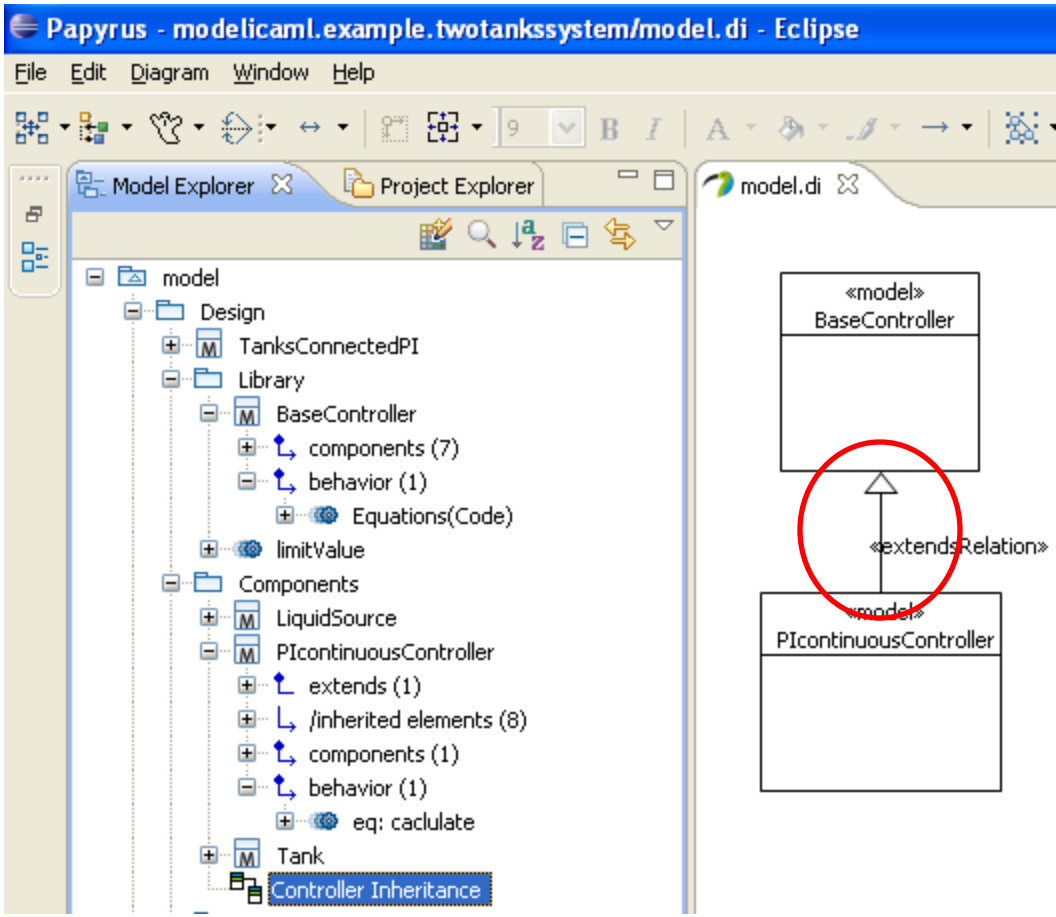


right click here…

Create Extends Relation

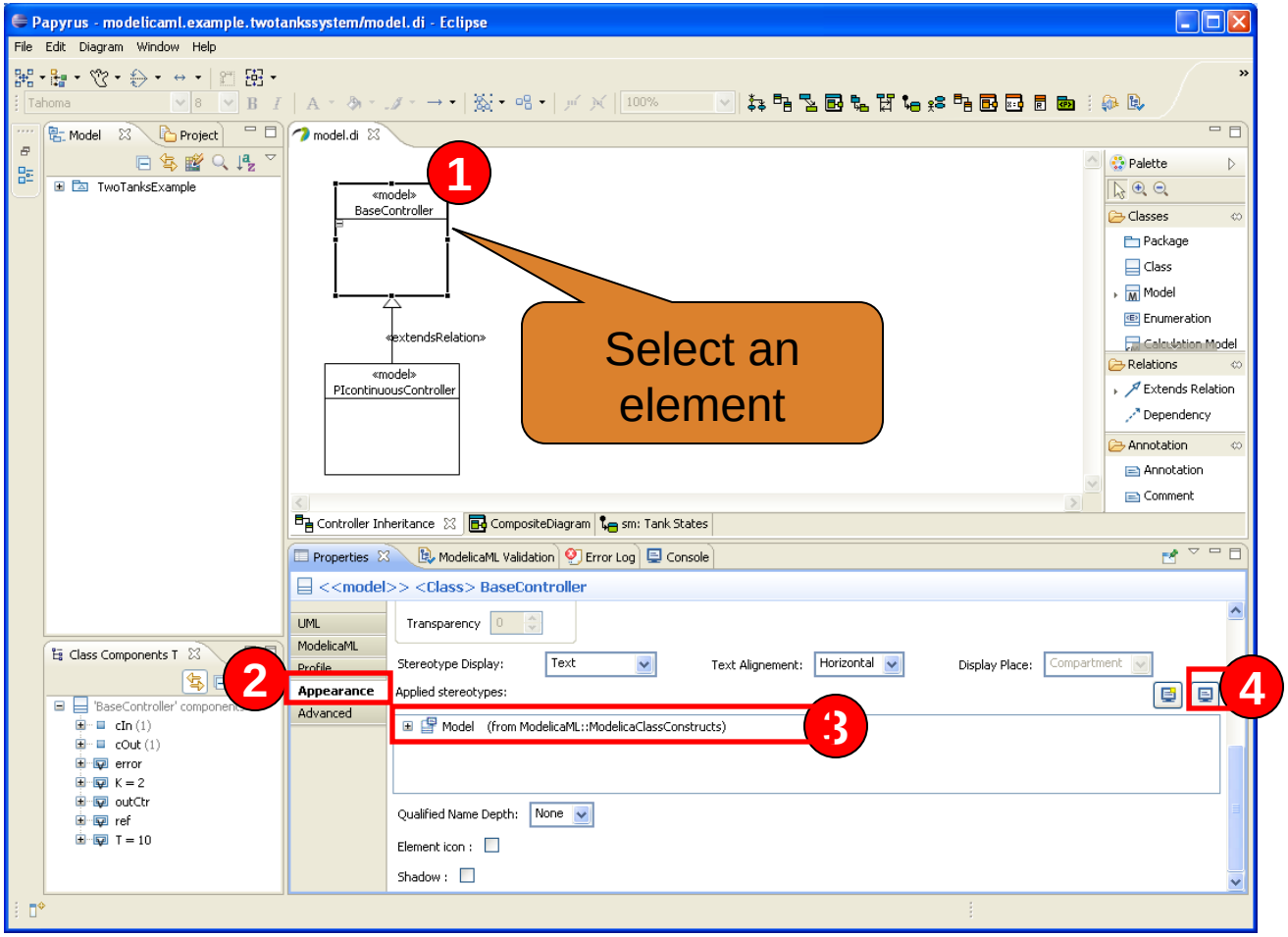
•Drag & drop BaseController and PIcontinuousController onto diagram

•Use the palette tool “Extends Relation”

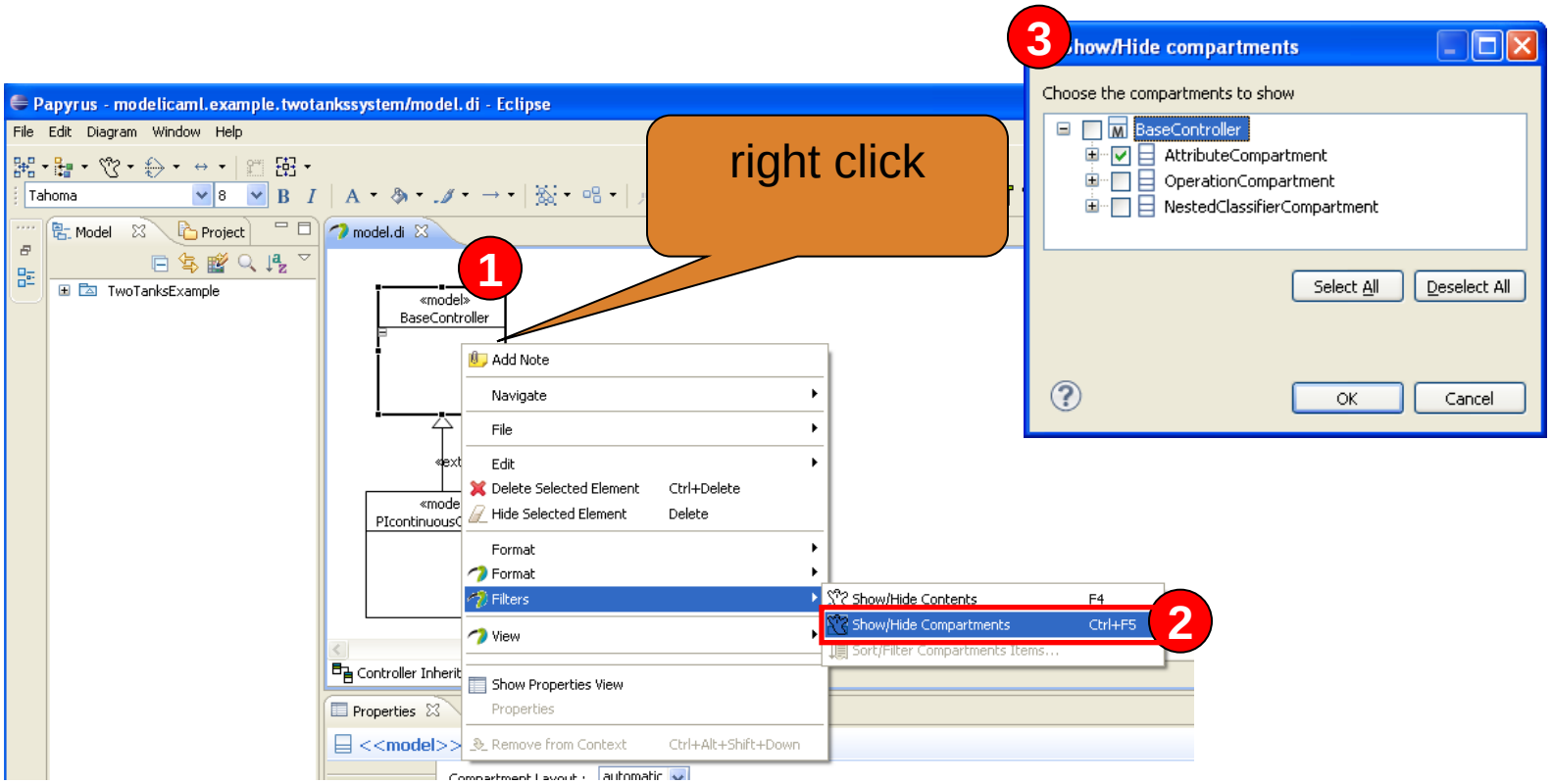


EADS

Hint: Element Appearance: Show stereotype name



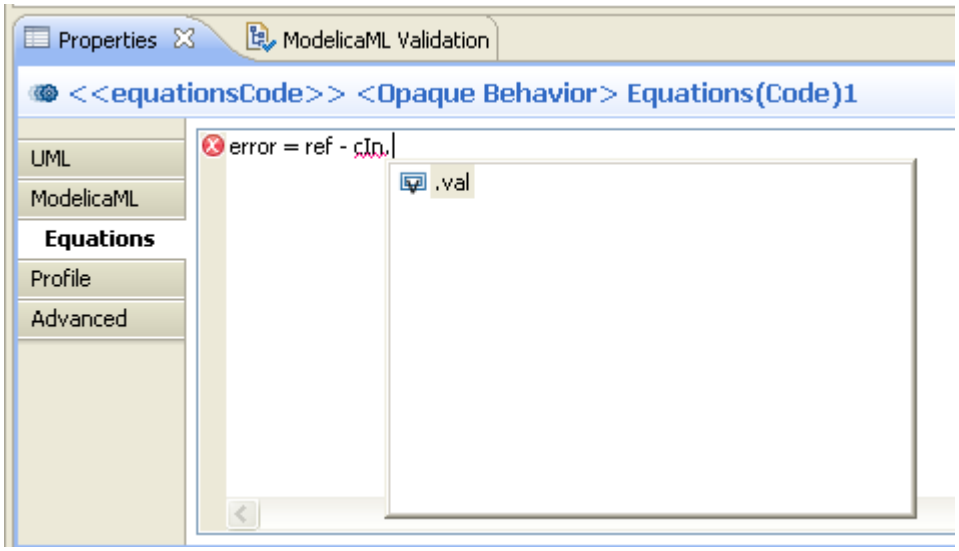
Hint: Element Appearance: Compartments



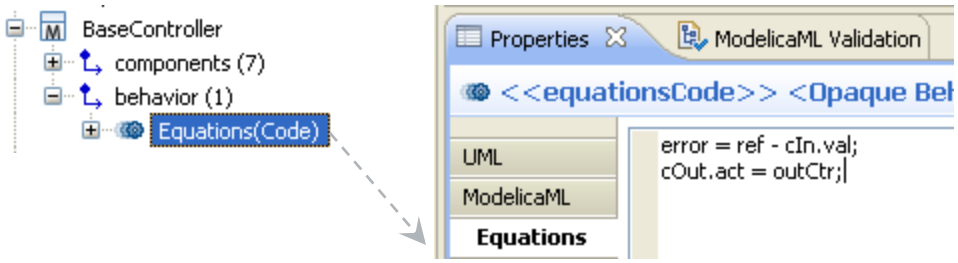
Behavior Modeling

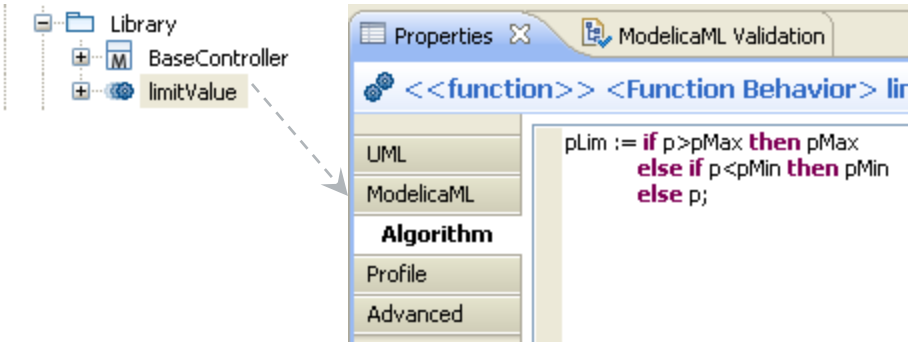
Hint: Editing Modelica Code

- Syntax highlighting and code completion is supported in code editors
- Hit **Ctrl + Space** for code completion when editing Modelica code

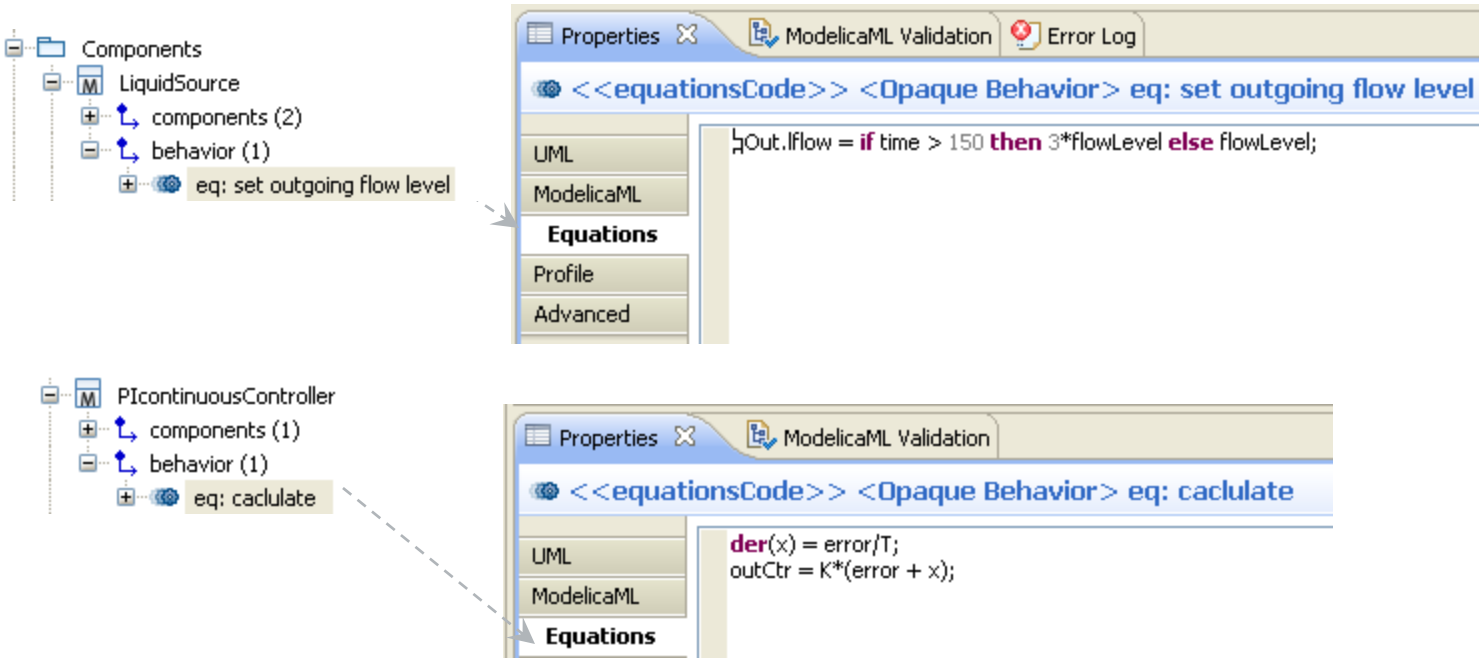


Create Behavior

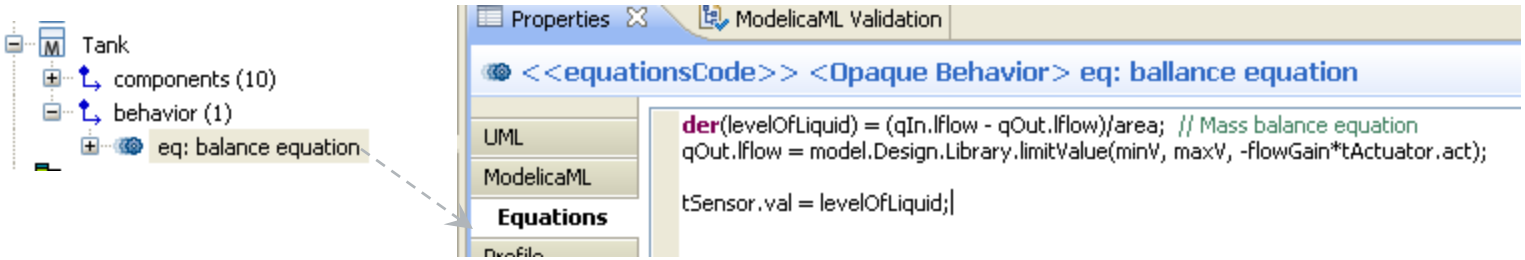




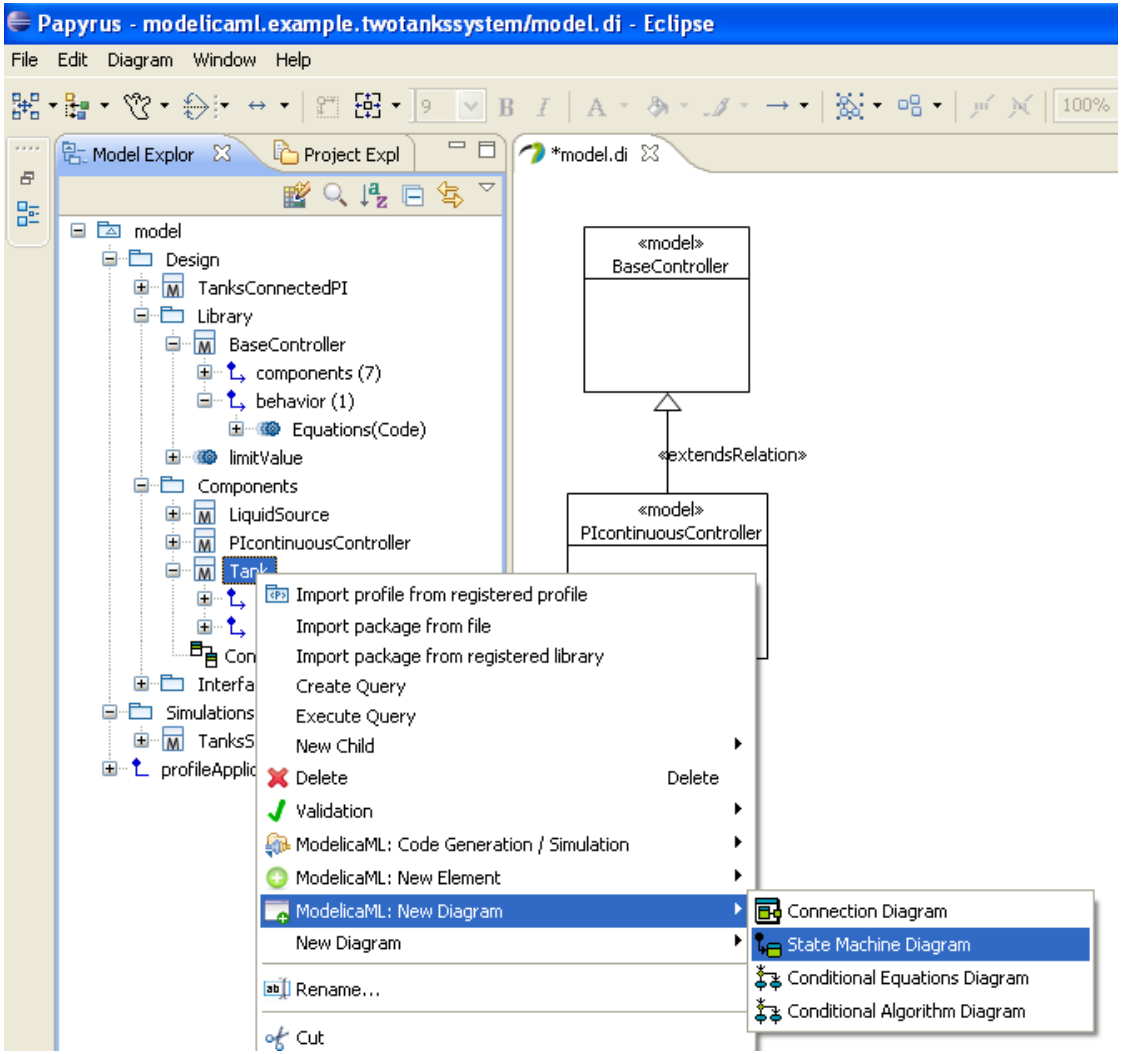
Create Behavior



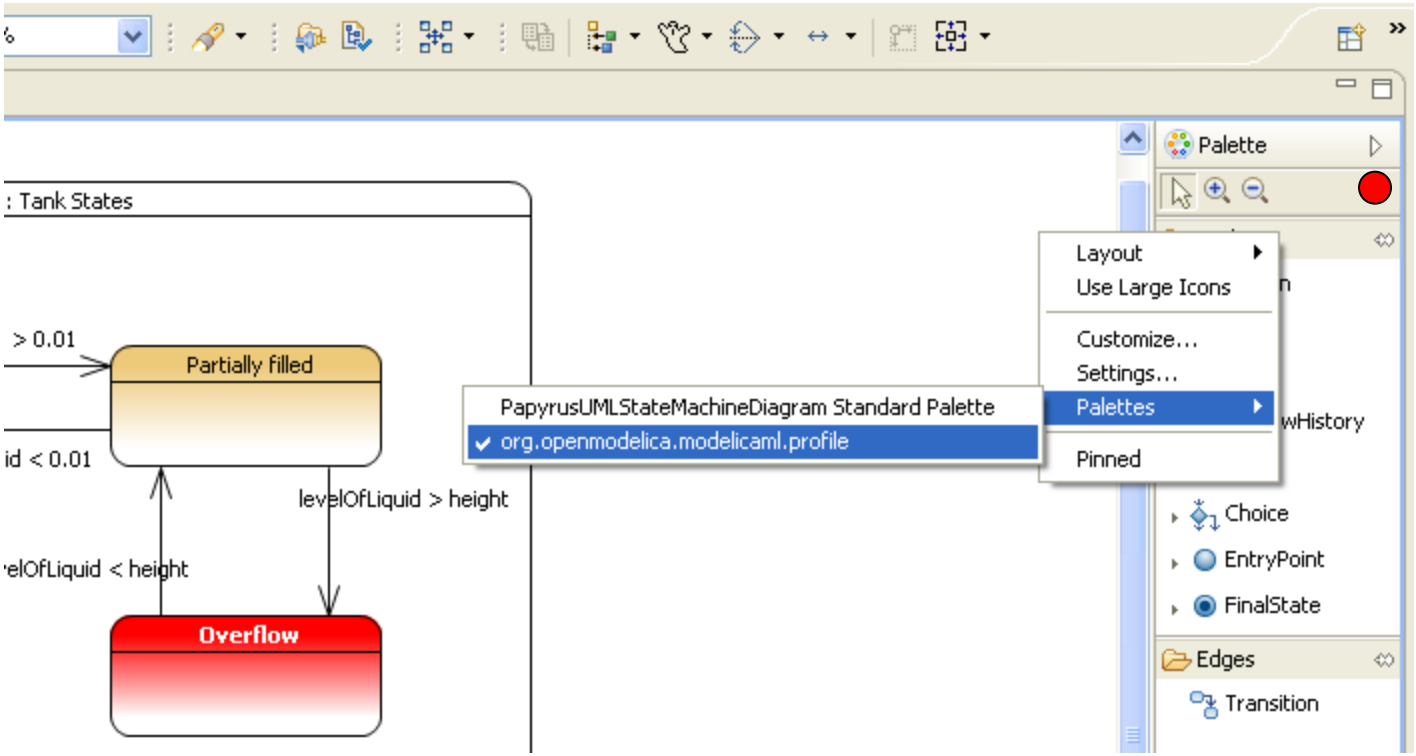
Create Behavior



Create State Machine



Configure Diagram Palette



right click here…

EADS

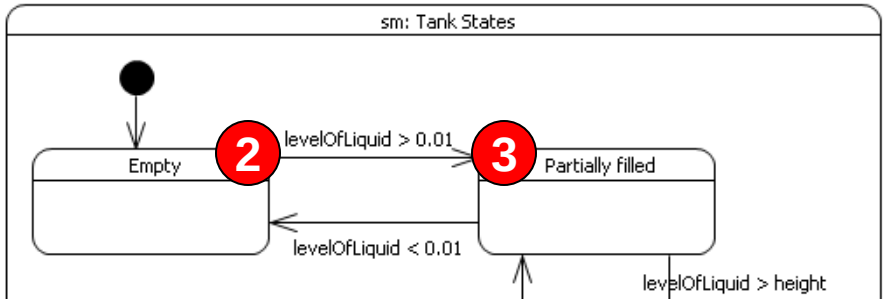
General: Working with diagrams

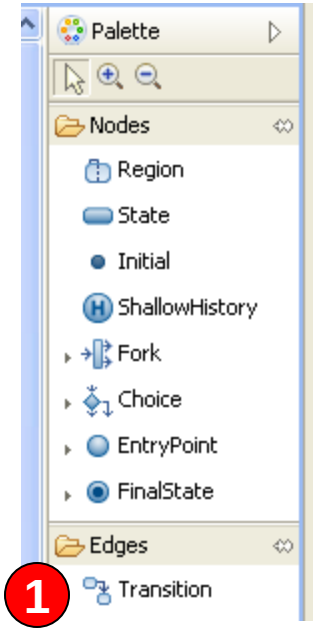
State Transitions:

1. Select the palette tool

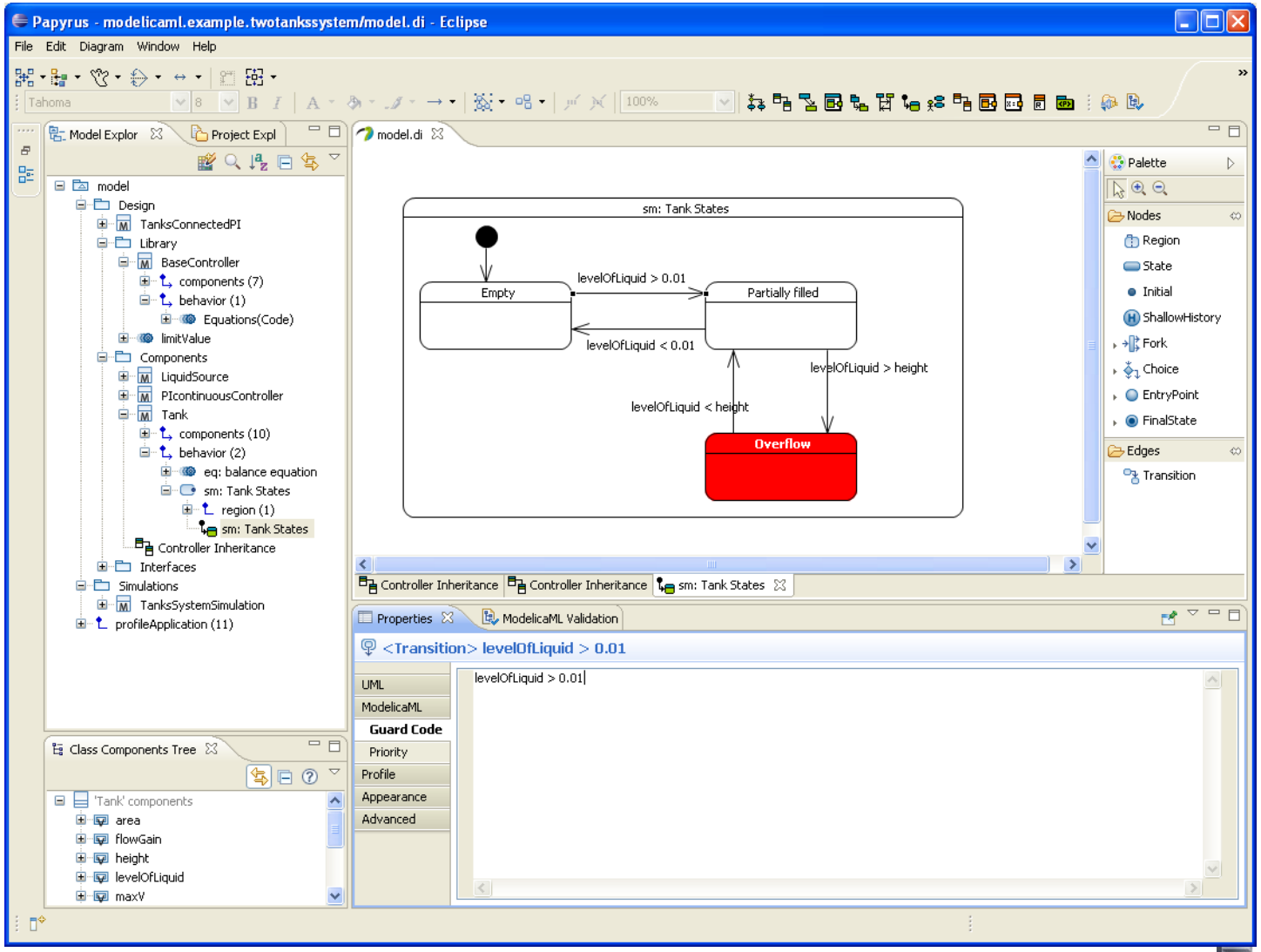
2. Click on the transition source state (click on label) and hold the mouse button

3. Move the mouse to the target state (to its label) and release the mouse button





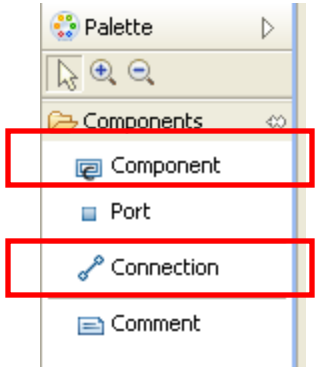
Create State Machine



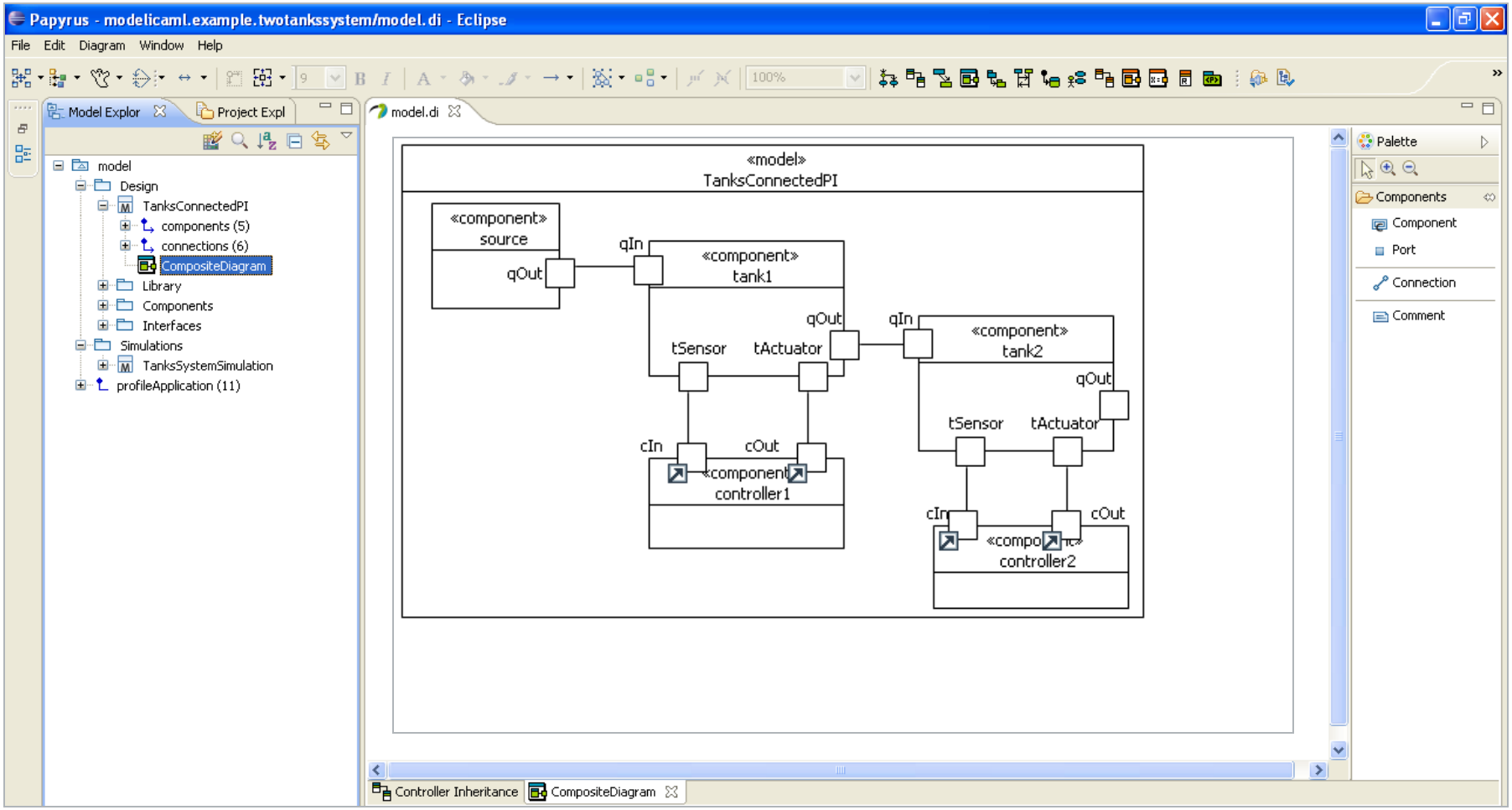
Architecture Modeling

Create Connection Diagram

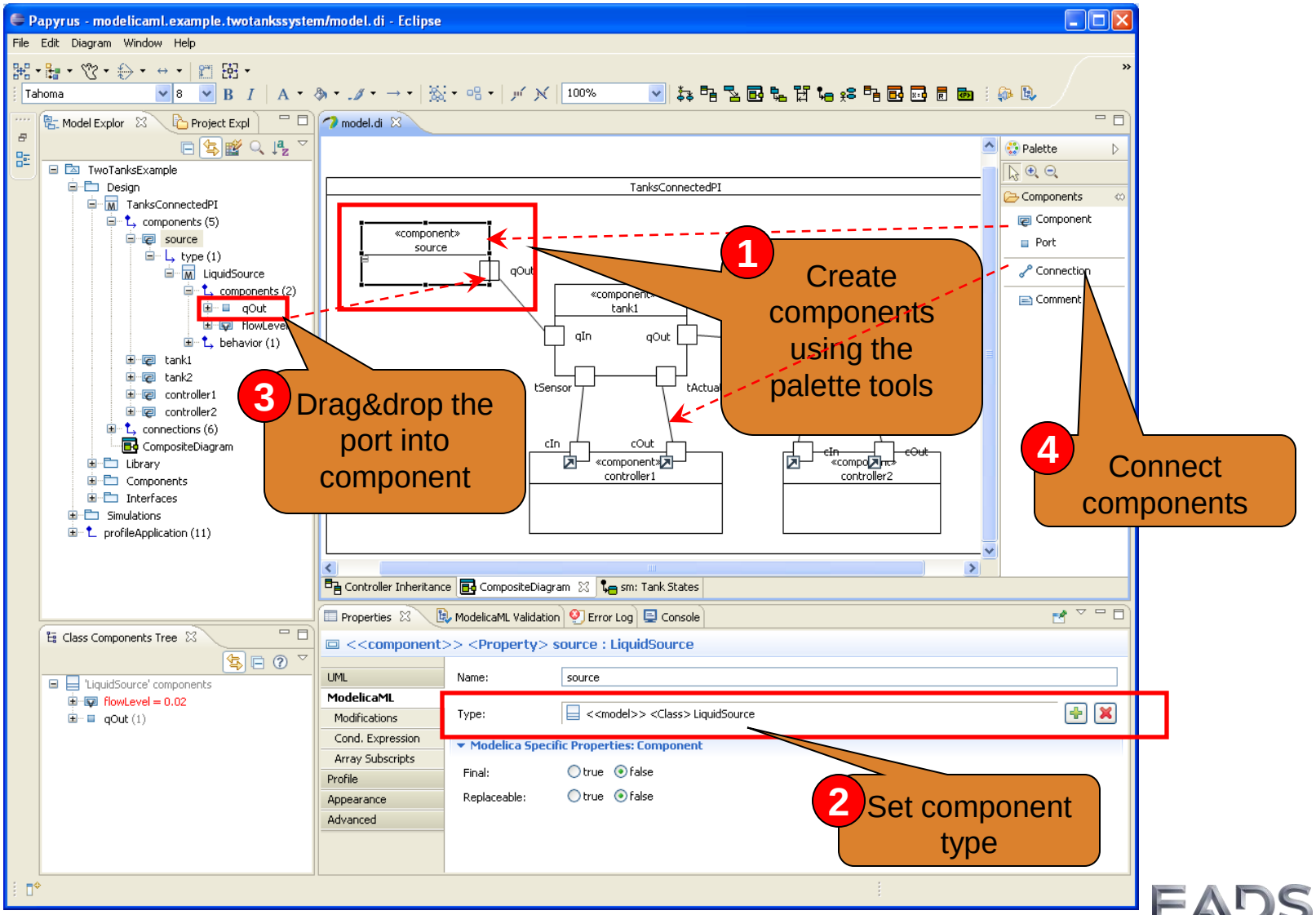
- Create a ModelicaML Connection Diagram under the TanksConnectedPI class
- Use components tool from the palette to create components inside the class on the diagram
- Define the types of components
- Use Model Explorer to find the ports
- Drag&Prop ports into respective components
- Arrange the components
- Use the “Connection” tool from the palette for connecting ports



Model System Architecture

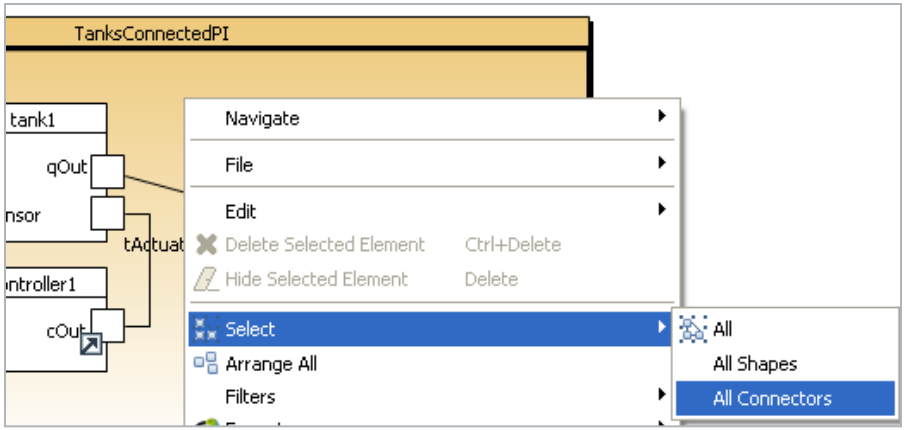


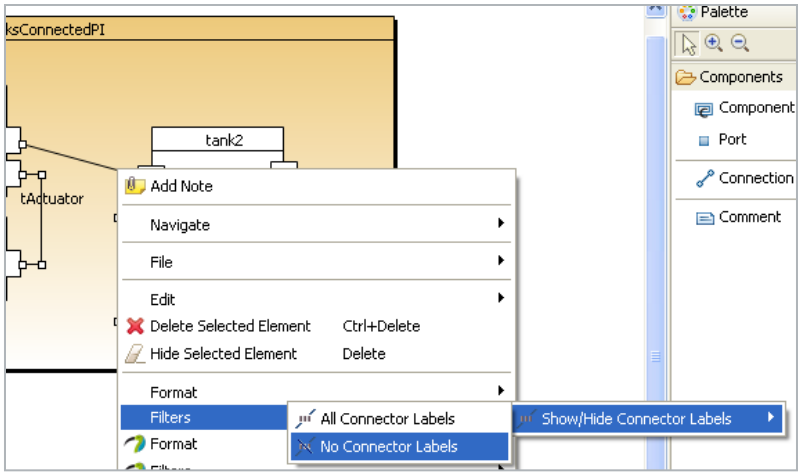
Create Connection Diagram



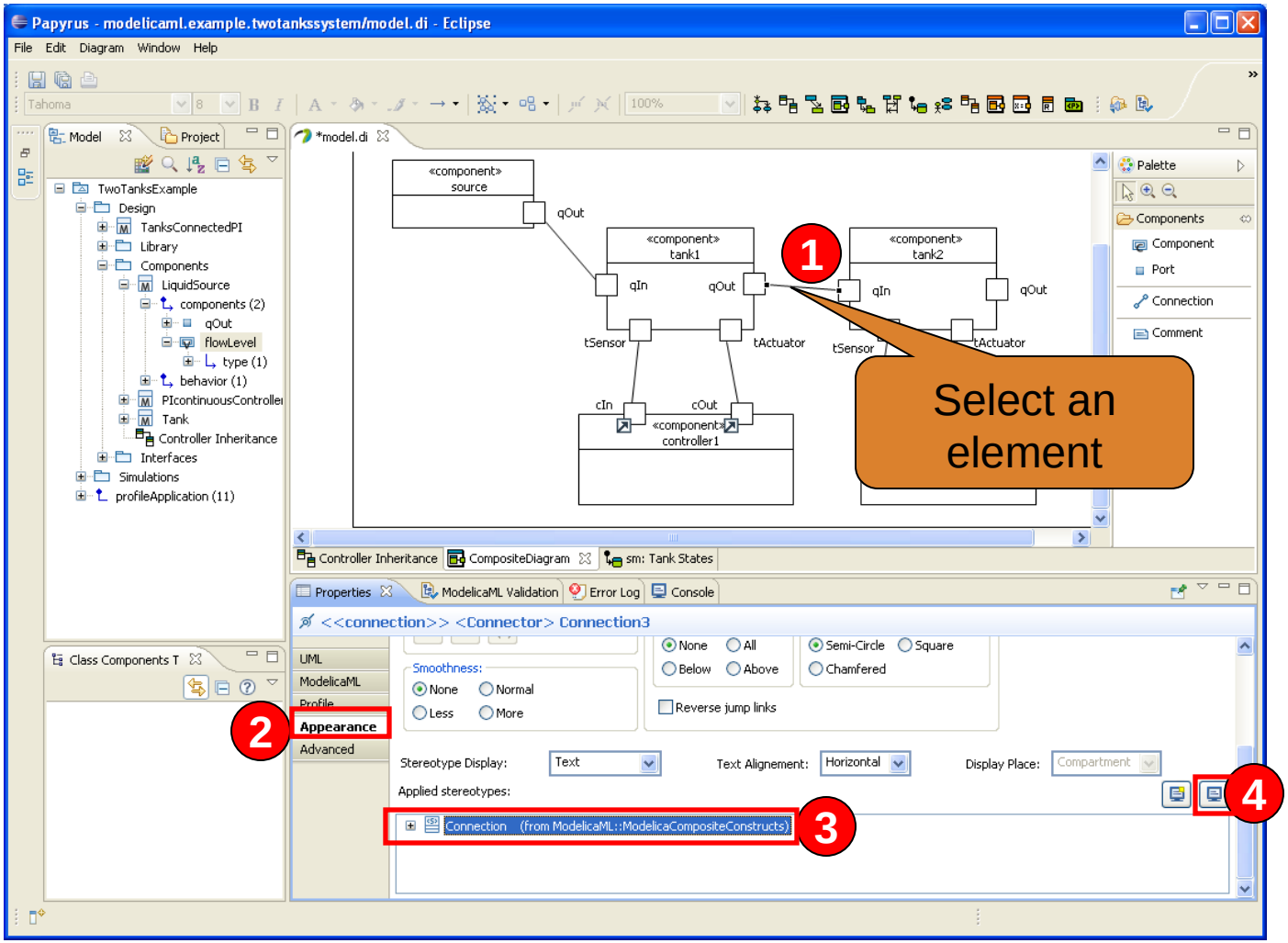
Hint: Hide the name of all connectors

- Click on the compartment of the class
- Right-click -> “Select” -> “All Connectors”
- Right-click <u>on one of the selected connectors</u> -> “Filters” -> “Show/Hide connector Labels” -> “No connector Labels”



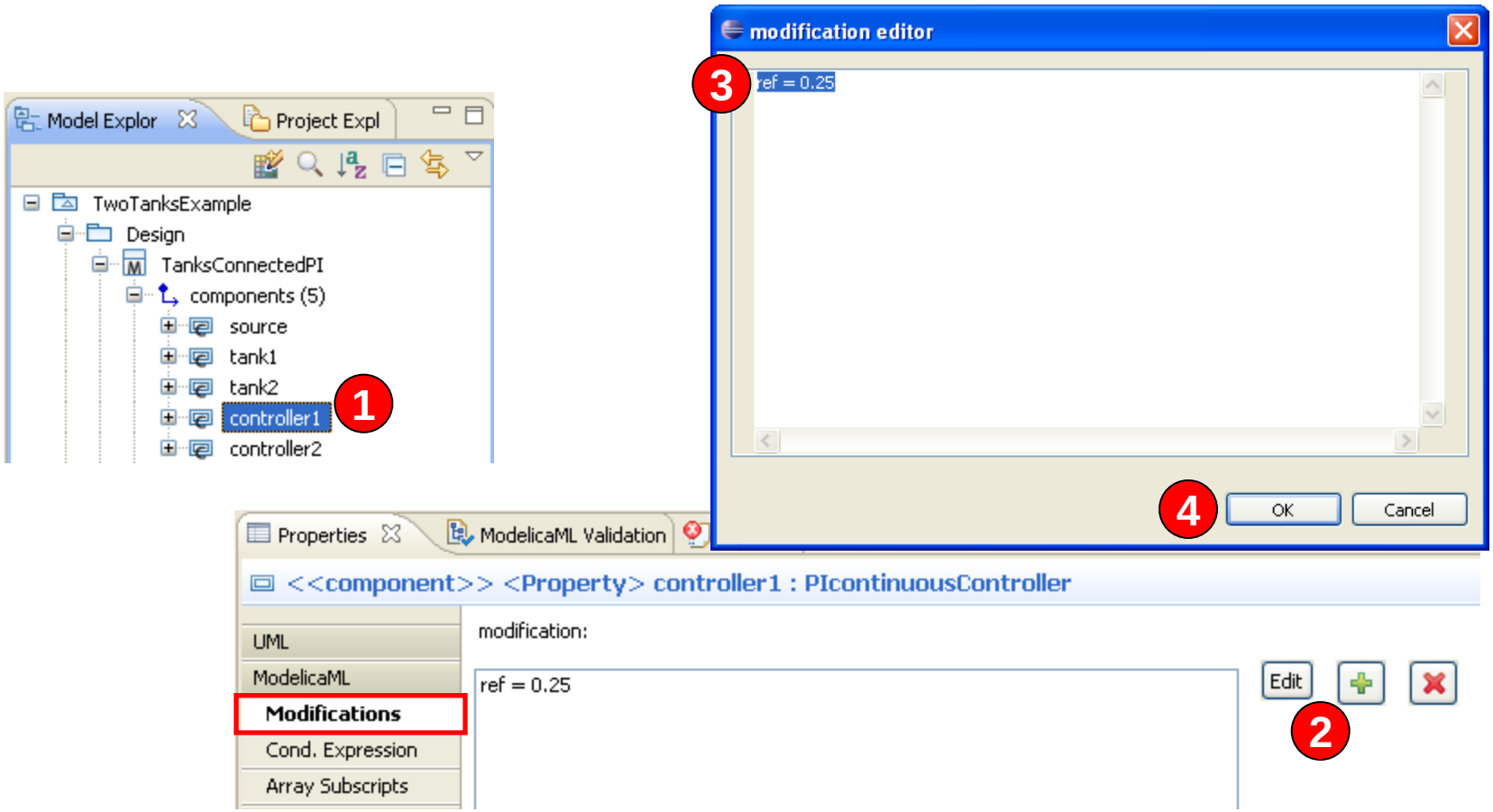


Hint: Element Appearance: Hide the name of the connection stereotype



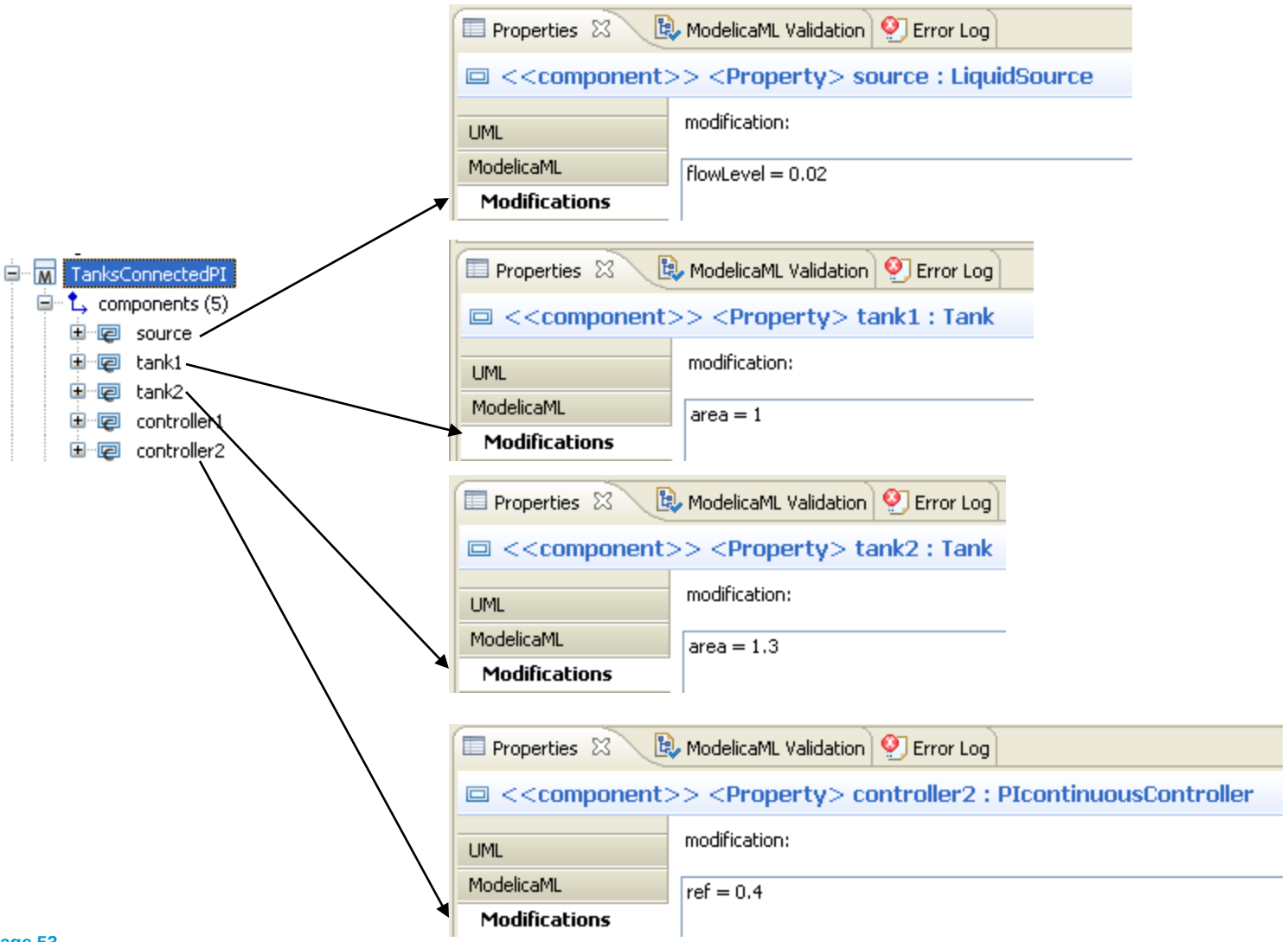
Component Modifications

Define Component Modifications



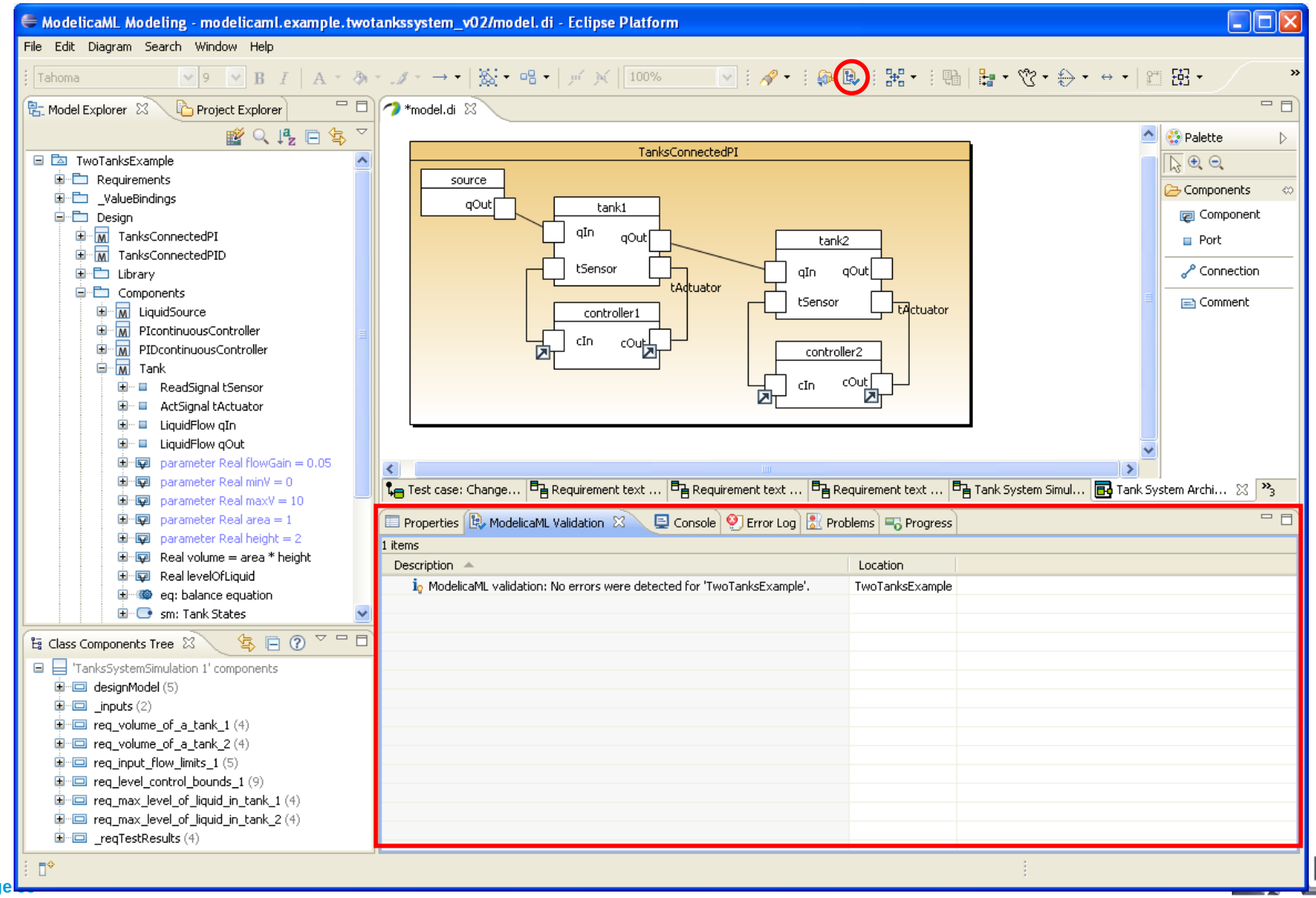
EADS

Define Component Modifications



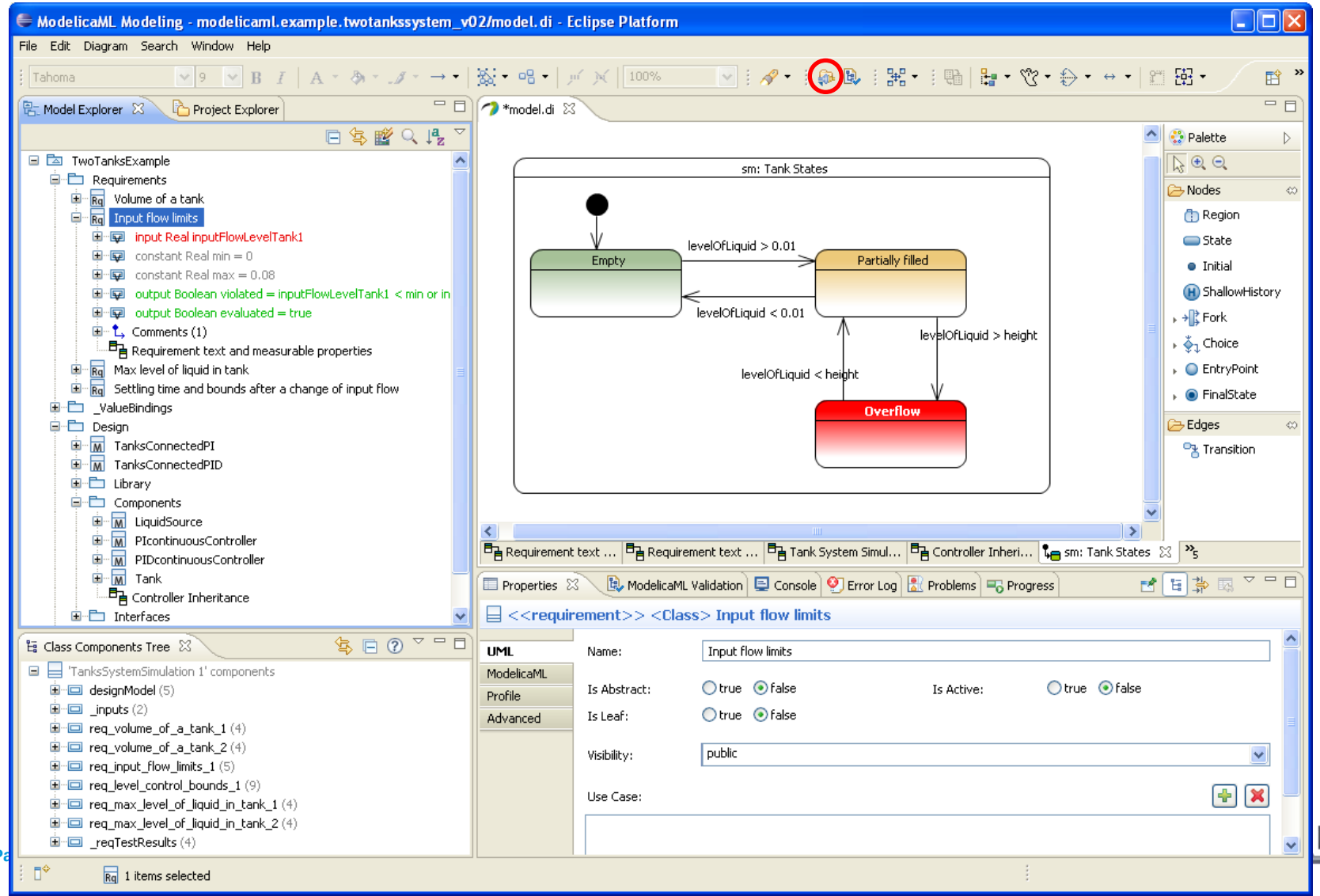
Model Validation

Validate Model



Modelica Code Generation

Launch Modelica Code Generation



Generated Modelica Code