

Tutorial 4 — Controller FMU Export (Overture)

Overview

This INTO-CPS tutorial will show you how to:

1. Generate a new controller FMU in Overture
 - (a) Import a model description into Overture
 - (b) Complete the skeleton model to produce a working controller
 - (c) Export the controller FMU
2. Associate the new controller FMU with a multi-model configuration
3. Execute a co-simulation using the new controller

Requirements

This tutorial requires the following tools from the INTO-CPS tool chain to be installed:

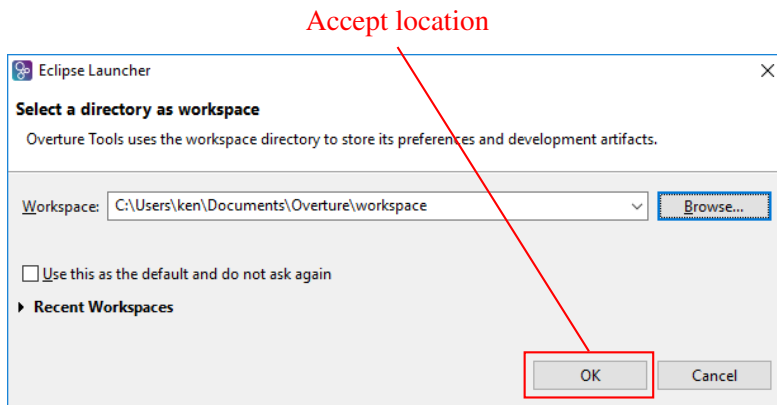
- INTO-CPS Application
- COE (Co-simulation Orchestration Engine) – accessible through the INTO-CPS App Download Manager
- Overture tool – accessible through the INTO-CPS App Download Manager
- Overture FMU Import/Exporter (No need for the CLI version) – accessible through the INTO-CPS App Download Manager

If you are following the tutorials for the first time at this point you need to install Overture and Overture FMU Import/Exporter through *Window > Show Download Manager* to your *into-cps-projects* install downloads directory (see the INTO-CPS Application *Window > Show Settings* menu). Please ask if you are unsure.

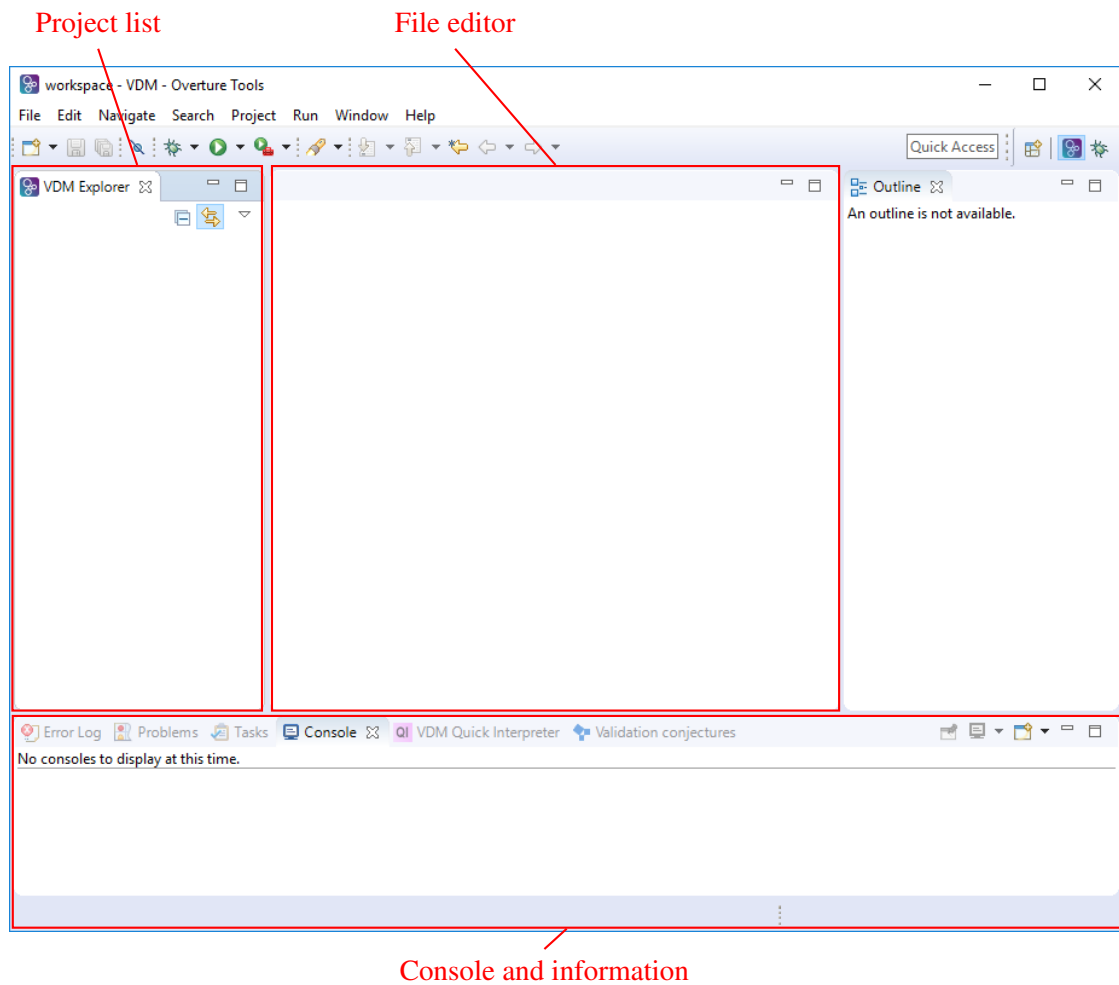
1 Creating a Project in Overture

The example in this tutorial is a small line-following robot with two infrared sensors. We will generate a controller FMU that reads these sensors and controls the wheels to follow the line. In the previous tutorials, this controller was already given as an FMU. First we will create a project.

Step 1. Open *Overture*. It will prompt you to select a location for its workspace. You may accept the default location by pressing *OK*, or press *Browse...* to select a different location. If you do not want to be prompted in future, check *Use this as the default and do not ask again*.

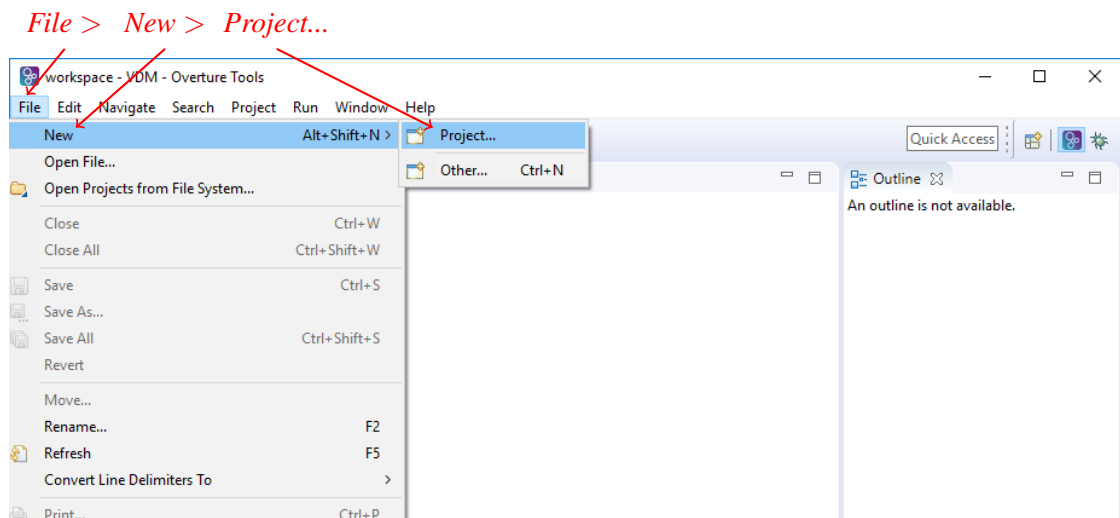


This is the *Overture* window, which includes a project list, file editor and a console.

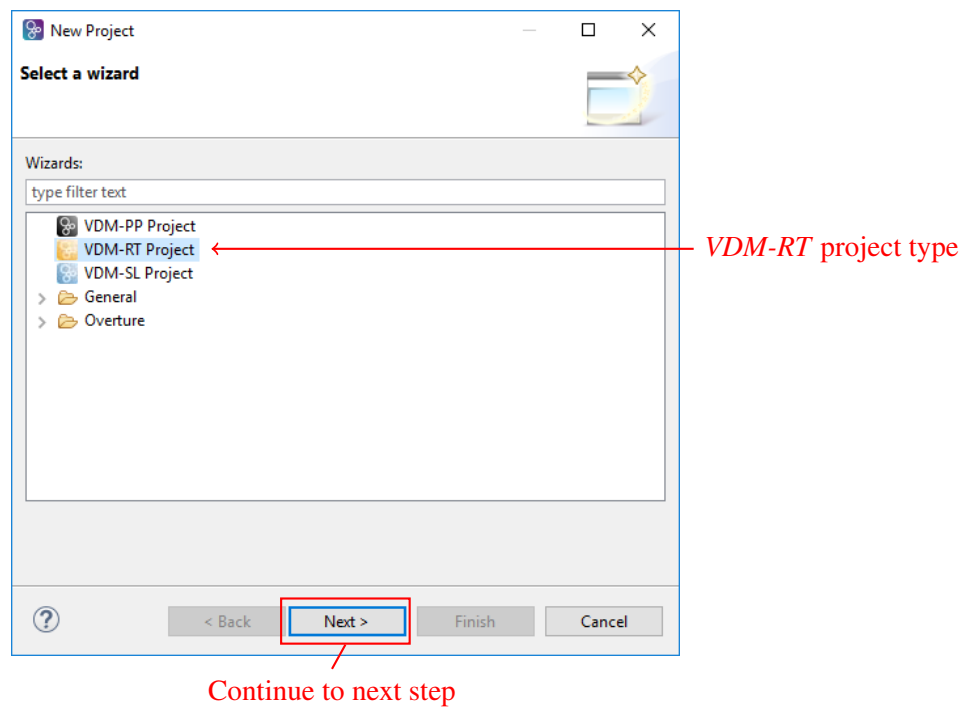


Remark. The INTO-CPS Application Download Manager will download Overture into the *Downloads Location* directory, and then will try to extract it to the *Downloads Installation Location*. Both these settings are in the INTO-CPS App Window > *Show Settings* menu. Sometimes, the operating system does not allow the INTO-CPS App to extract the downloaded archive, so you have to perform this action manually. If you encounter errors opening Overture, they could be caused by a partial extraction. In that case, remove Overture's installation folder and extract the files manually.

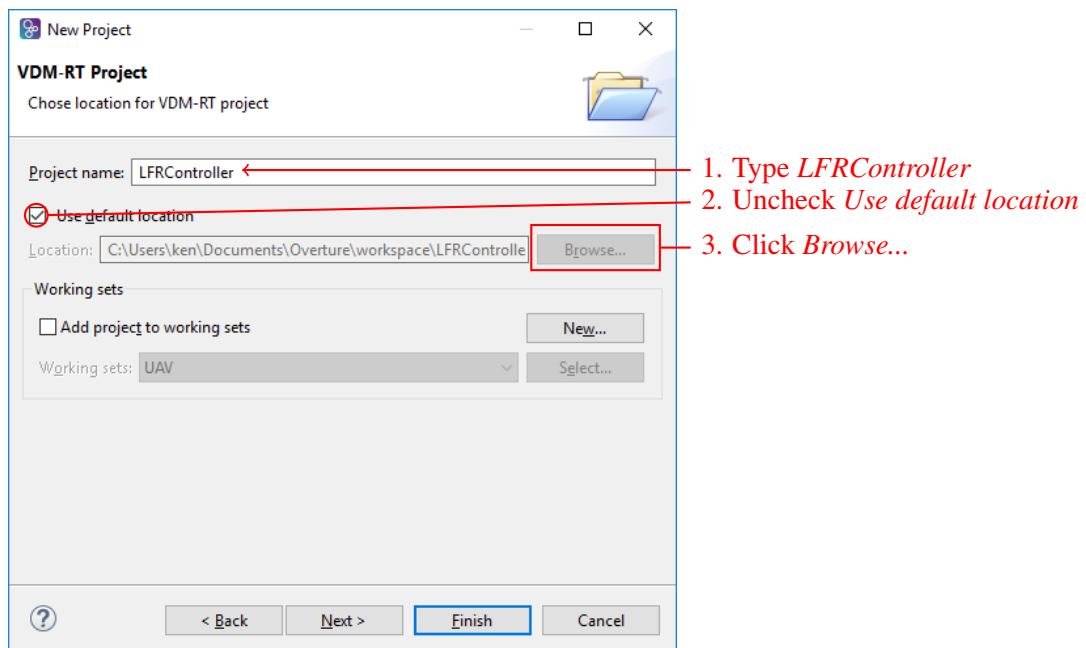
Step 2. First create a project that will hold the controller model. Select *File > New > Project...*



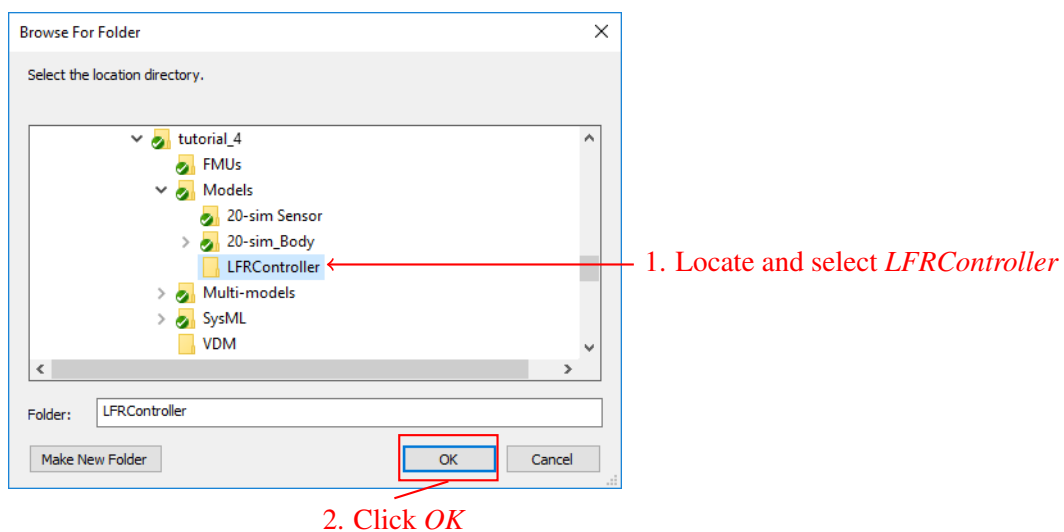
Step 3. In the *New Project* window, select *VDM-RT Project* and click *Next >* to go to the next step.



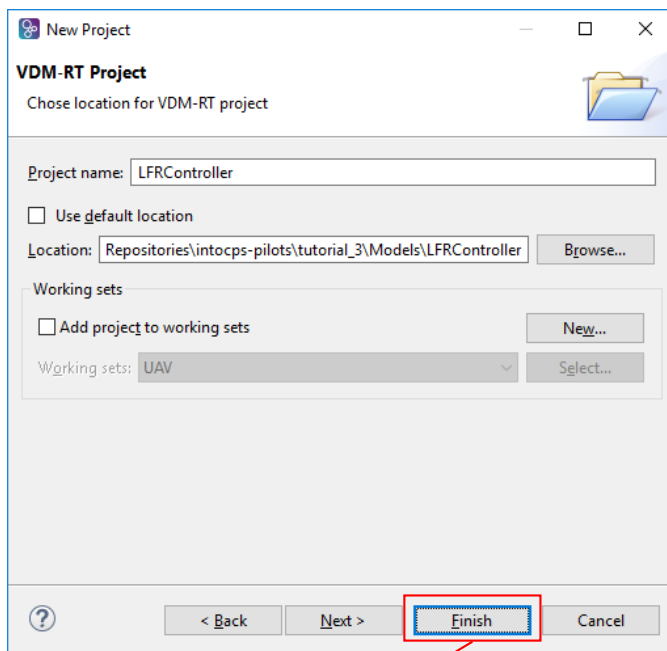
Step 4. The next screen asks for a name for the project. Call it *LFRController*. We will place the project in the *tutorial_4/Models* folder, so uncheck *Use default location* and click *Browse...*



Step 5. Locate and select the folder *tutorial_4/Models/LFRController* and click *OK*.

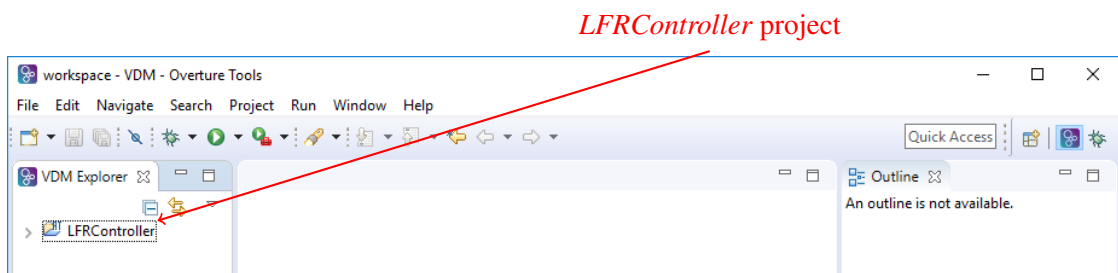


Step 6. Finally click *Finish* to create the project.



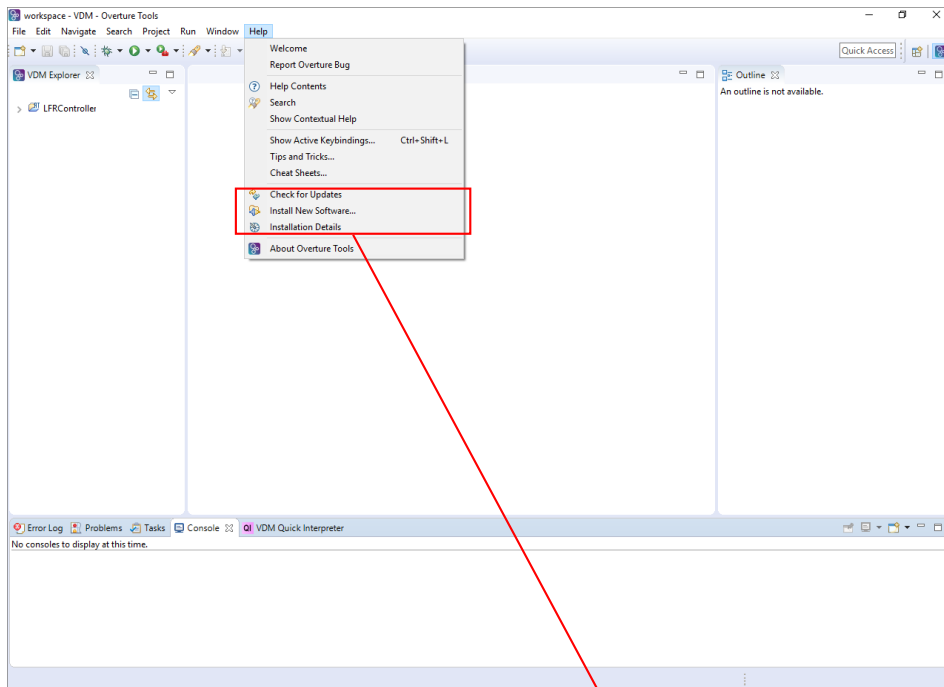
2. Click *Finish*

You should see the new project in the project list.



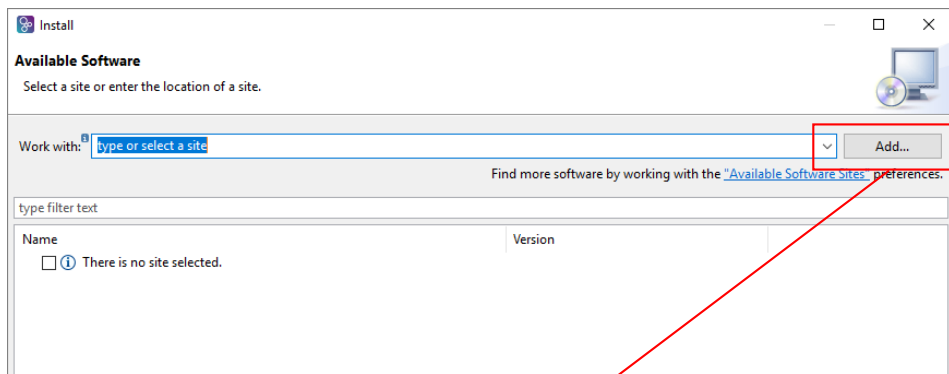
2 Installing the Overture FMU plug-in

Step 7. Select *Help > Install New Software...*



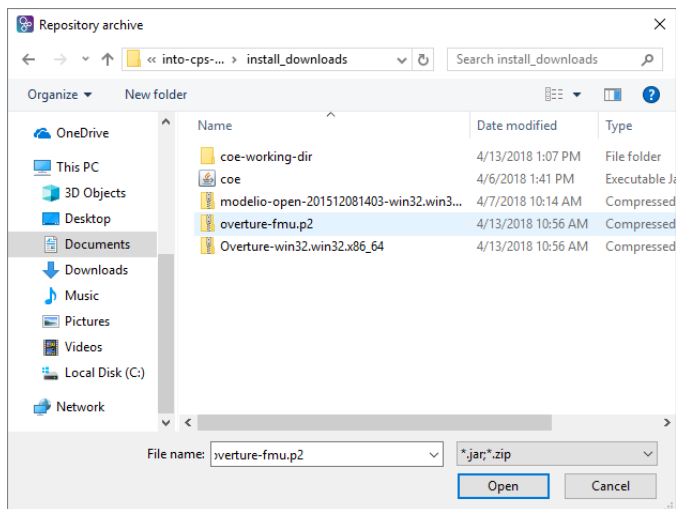
Click Install New Software...

Step 8. Select *Add* and then *Archive...*



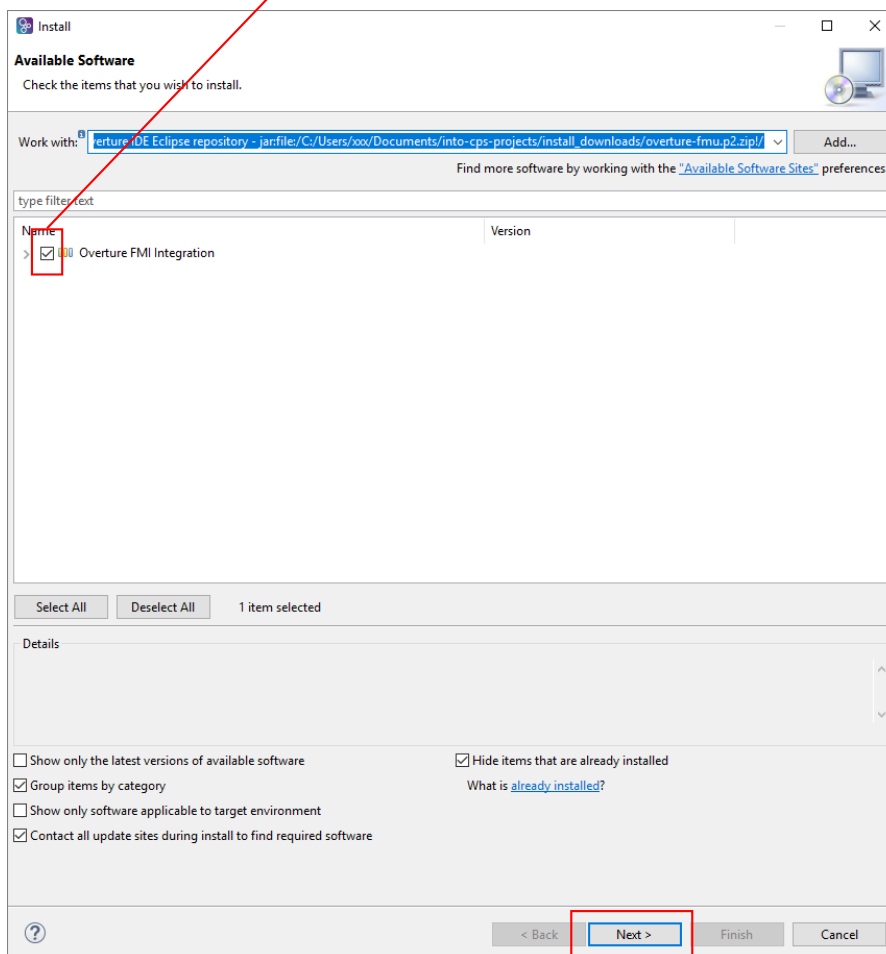
Click Add...

Step 9. Open the `overture-fmu.p2` file inside your *into-cps-projects/install_downloads* folder.



Step 10. Select the plugin and click Next to continue the installation. When finished Overture restarts and you are ready to go to the next step.

Check Overture FMI Integration



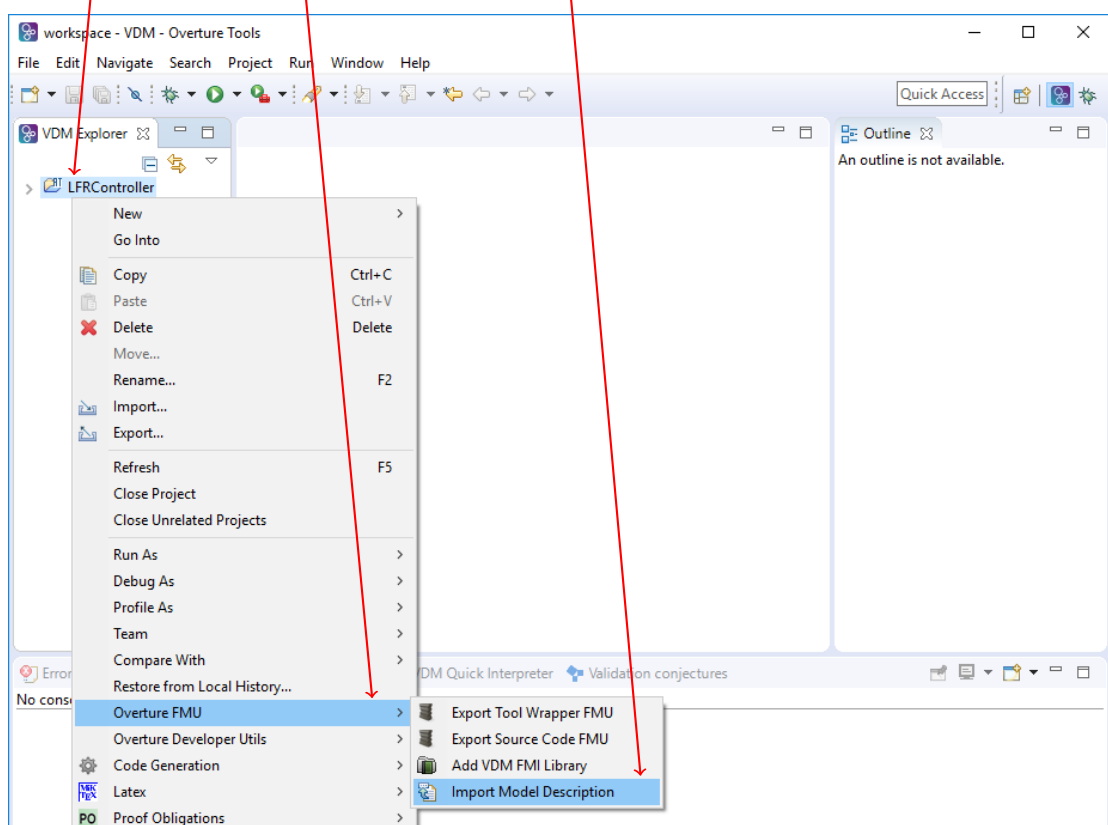
Next

3 Importing a Model Description into Overture

Overture can import model description files to create a skeleton project with the correct input, output and parameter ports, as well as standard boilerplate elements needed in a VDM-RT model.

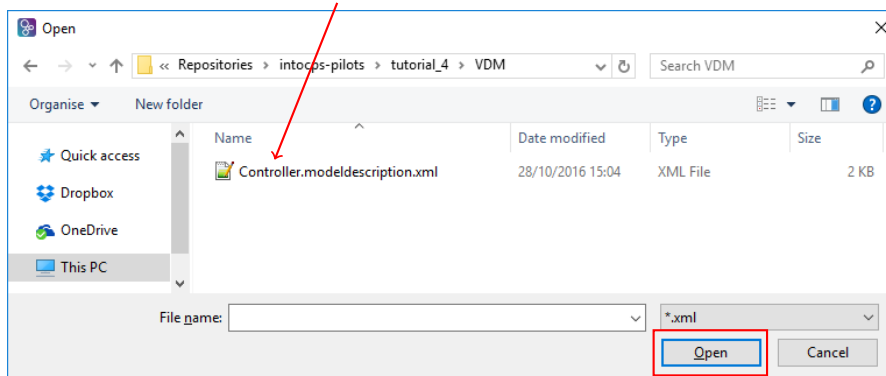
Step 11. To import a model description, right-click on the *LFRController* project and select *Overture FMU > Import Model Description*.

Right-click... *Overture FMU > Import Model Description*



Step 12. Locate the file *tutorial_4/VDM/Controller.modeldescription.xml* that is included in the project and click *Open*.

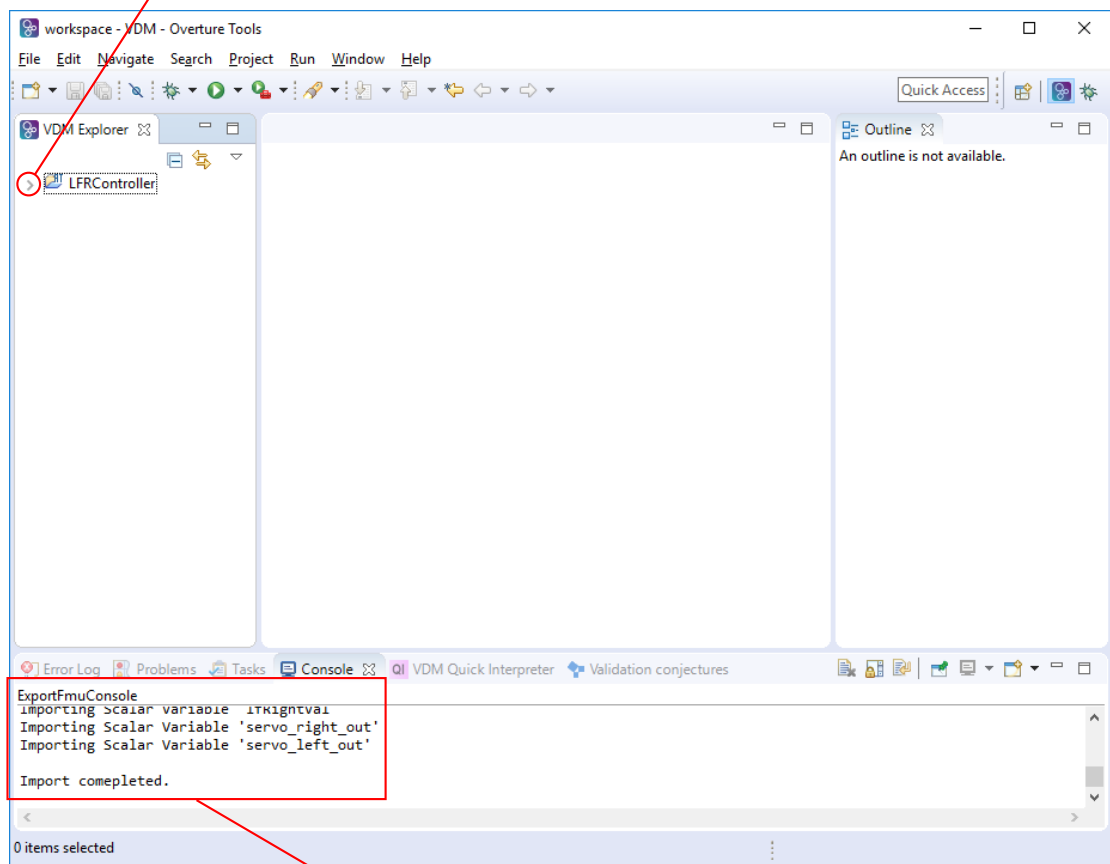
Locate and select *Controller.modeldescription.xml*



2. Click Open

Step 13. Overture will parse the file and populate the project. You can see status messages from the import in the *Console*. Expand the *LFRController* project to see what was imported.

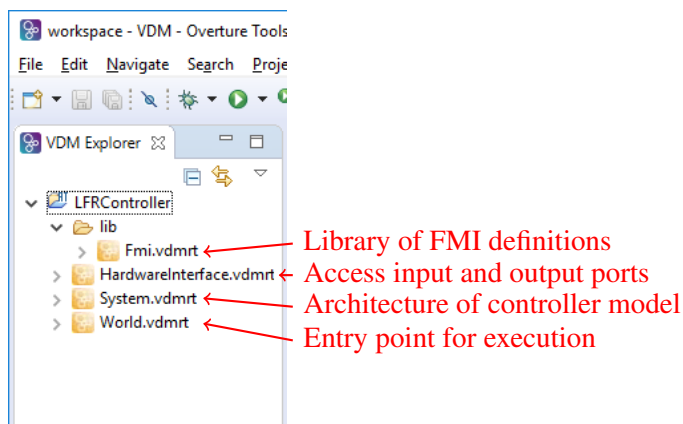
Expand *LFRController*



Import status

Remark. Sometimes this step fails. If you don't see the "Import completed" message, remove all imported files and retry the current step.

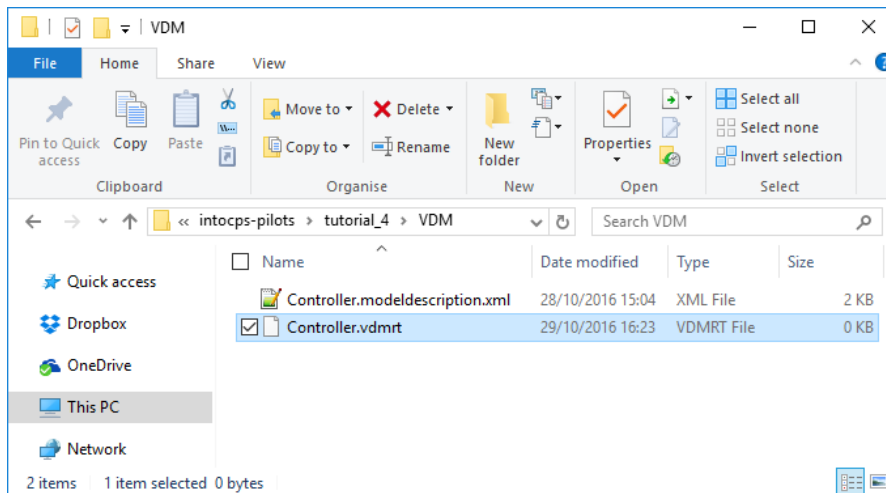
You should see the following structure:



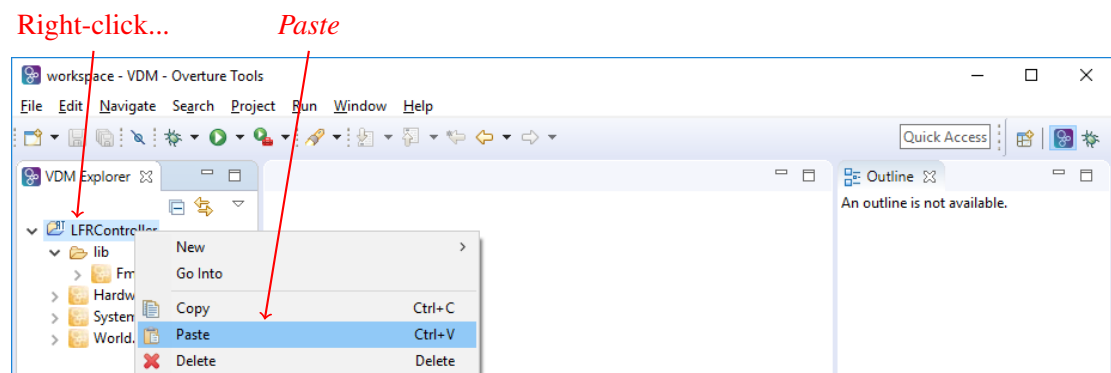
4 Adding a Controller Class

To make a functional controller, we will add a *Controller* class and instantiate it as an object in the *System* class, and set the *World* to start the controller thread. A basic controller class is included in the *tutorial_4* project.

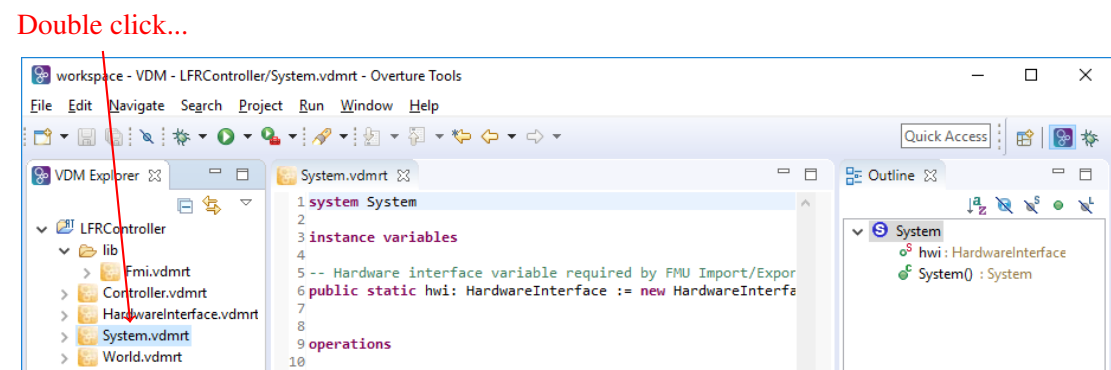
Step 14. Locate the file *tutorial_4/VDM/Controller.vdmrt* on on your file system and copy it.



Step 15. Right-click on the *LFRController* project and select *Paste*.



Step 16. Double-click *System.vdmrt* to open the *System* class.



Step 17. Add the highlighted lines to *System.vdmrt*. This will define a controller object of the Controller class and instantiate it.

```
system System

instance variables

-- Hardware interface variable required by FMU Import/Export
public static hwi: HardwareInterface := new HardwareInterface();

public static controller: Controller := new Controller(
    hwi.servoLeftVal, hwi.servoRightVal, hwi.lfRightVal, hwi.lfLeftVal);

cpu : CPU := new CPU(<FP>, 1E6);

operations

public System : () ==> System
System () ==
(
    cpu.deploy(controller);
);

end System
```

Step 18. Double-click *World.vdmrt* to open the World class. Uncomment the highlighted line to tell the controller thread to start at the beginning of co-simulation.

```
class World

operations

public run : () ==> ()
run () ==
(
    start(System\controller);
    block();
);

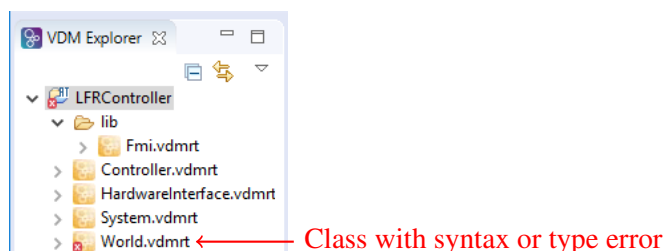
private block : () ==> ()
block () ==
    skip;

sync

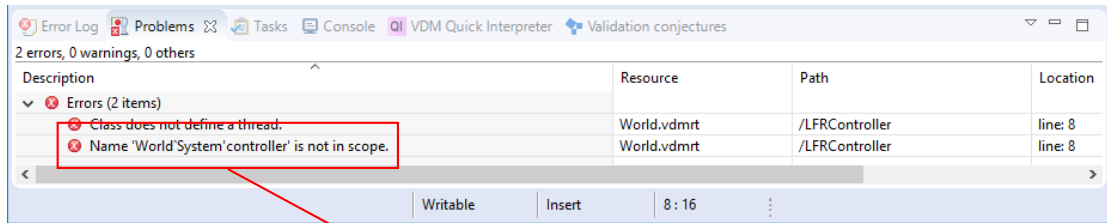
    per block => false;

end World
```

Step 19. Ensure that your model has no errors. If it does, a red cross will appear next to the file icon in the project browser. (You might have to refresh the project by right-clicking and selecting *Refresh* to see these.)



Check that you have correctly replicated the listings from Steps 13 and 14. Look at the *Problems* tab at the bottom for information, and double-click items to take you to the problem in the file editor.



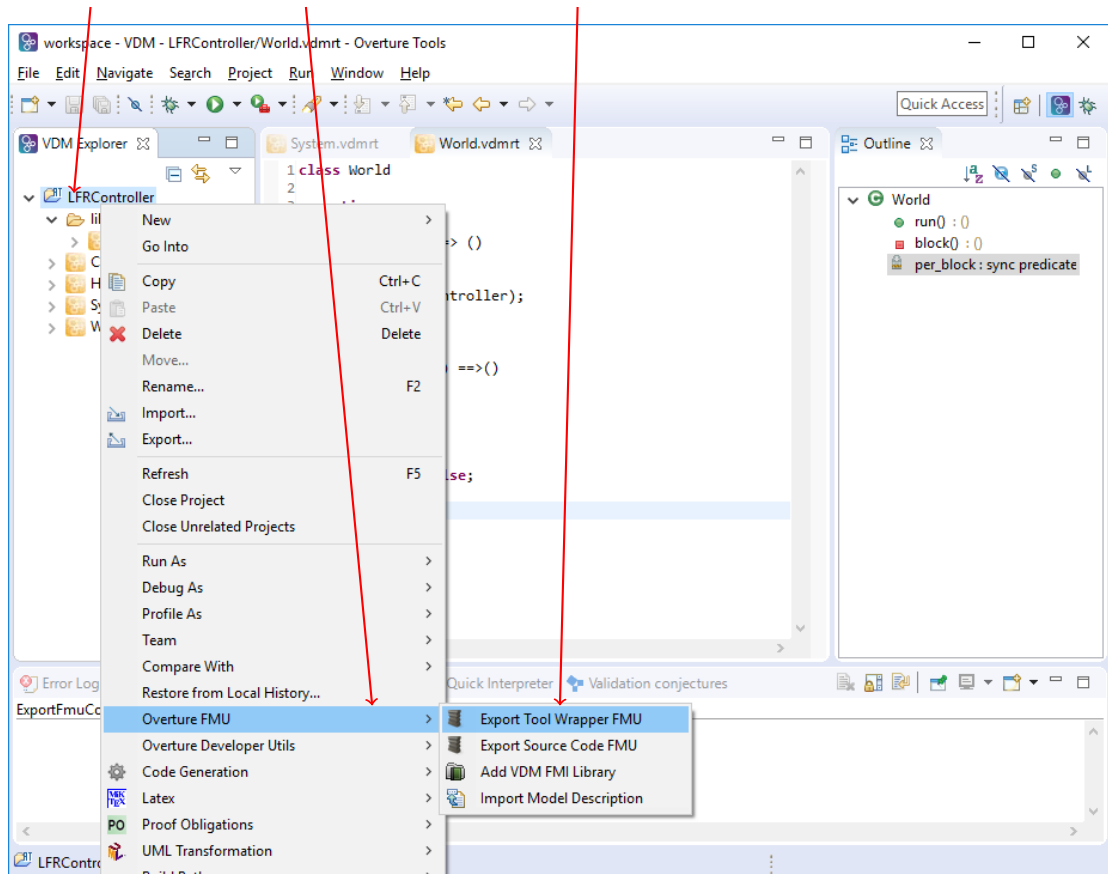
Double-click to go to the problem

5 Exporting an FMU and Adding it to a Multi-model

Now that the controller model is complete, we can export an FMU and place it in the *tutorial_4* where the INTO-CPS Application can see it.

Step 20. To export an FMU, right-click on the *LFRController* project and select *Overture FMU > Export Tool Wrapper FMU*.

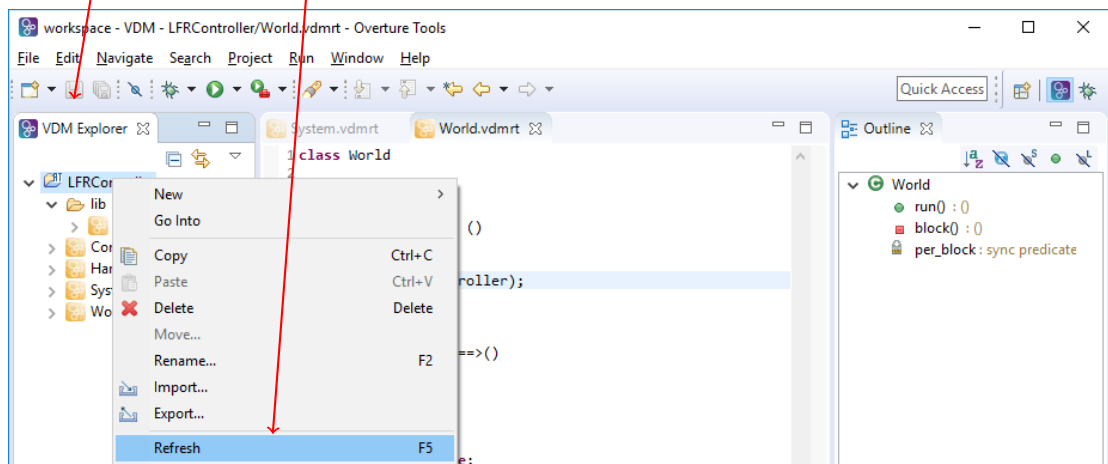
Right-click... *Overture FMU > Export Tool Wrapper FMU*



Step 21. Refresh the project so that the generated FMU appears. To do this, right-click on the project and select *Refresh*.

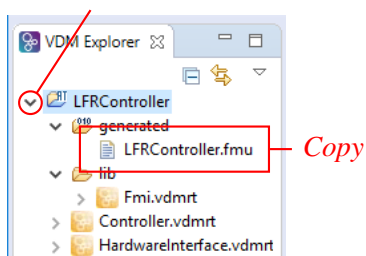
Right-click...

Refresh

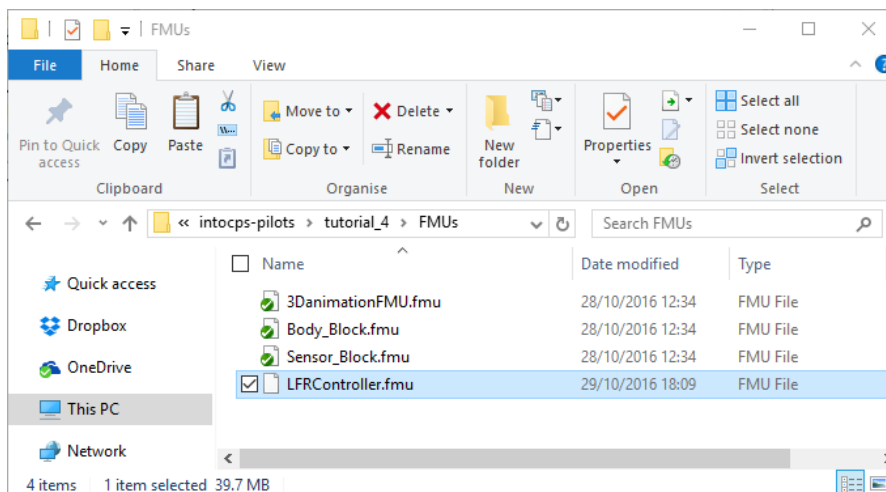


Step 22. A new folder called *generated* will appear. Expand this to see *LFRController.fmu*. Select *LFRController.fmu* and copy it using *Ctrl+C* or right-clicking and selecting *Copy*.

Right-click...



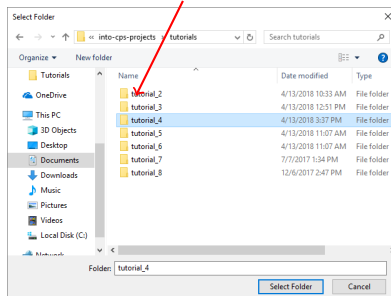
Step 23. Paste *LFRController.fmu* into the *tutorial_4/FMUs* folder on your file system.



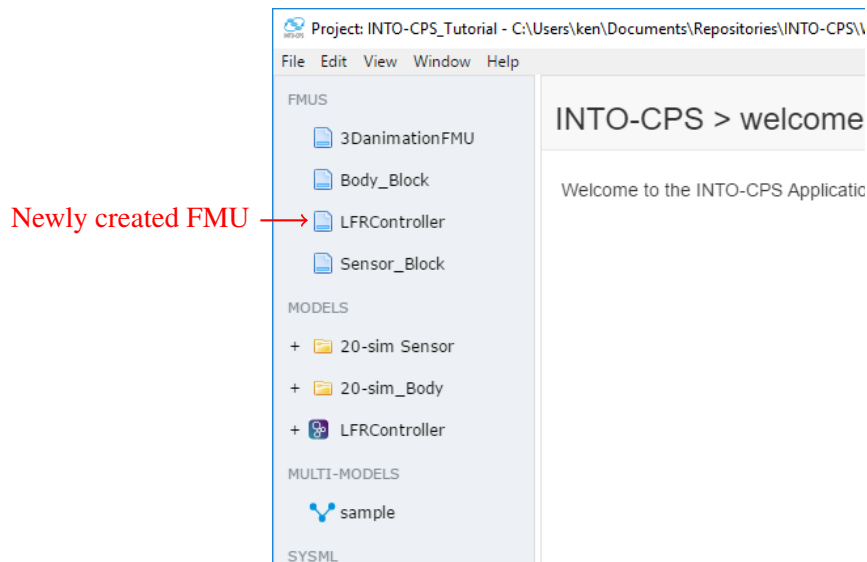
6 Co-simulating with the New Controller

Step 24. Launch the *INTO-CPS Application* and select *File > Open Project*. Set the *Project root path* to the location of *Tutorials/tutorials_4* and click *Open*. You can browse using the *Folder* button.

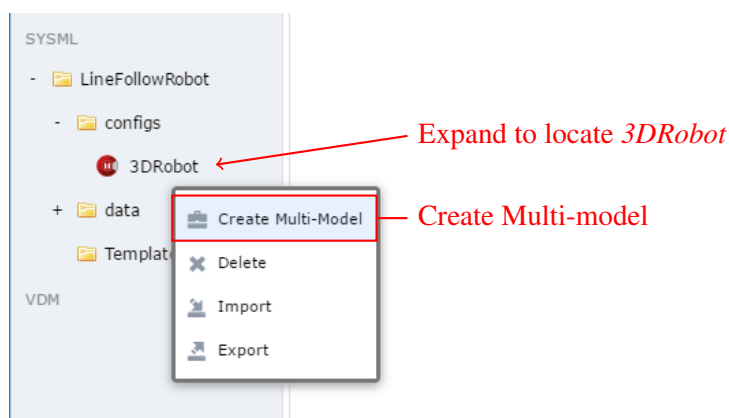
Path to Tutorials/tutorials_4



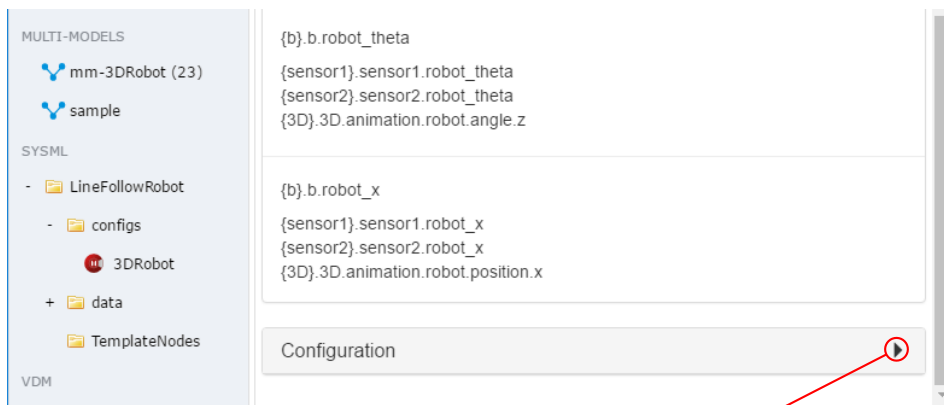
You should see the newly export *LFRController* FMU in the list.



Step 25. In the SysML entry of the project browser, expand the *LineFollowRobot* folder, then *config* folders. Right-click on *3DRobot* and select *Create Multi-Model*.



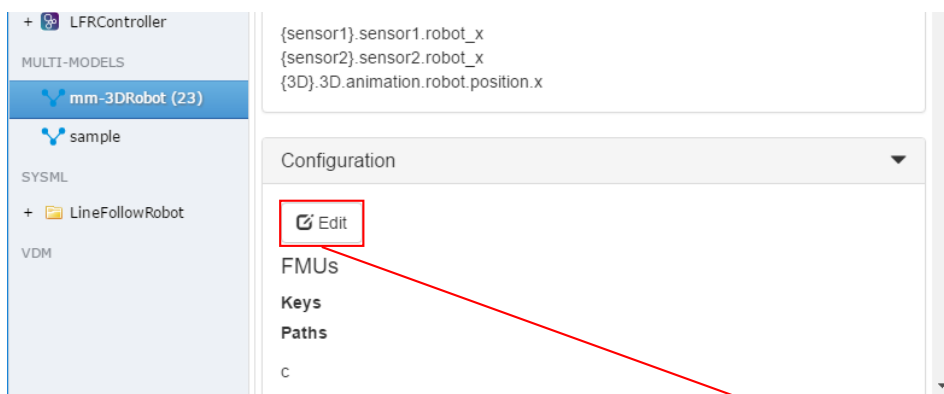
Step 26. We now need to associate FMUs to the multi-model as we did in *Tutorial 2*. Scroll down to find the *Configuration* panel and expand it by clicking the arrow.



Expand Configuration

Remark. If you are not on Windows, delete the *3D* FMU from the multimodel.

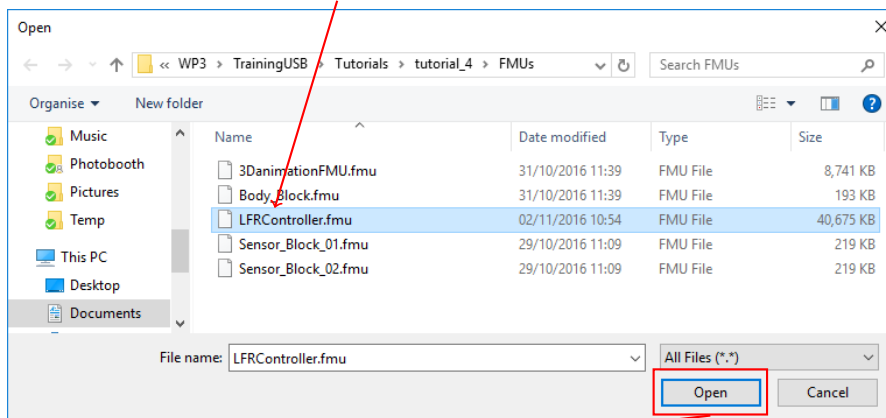
Step 27. Scroll down and click *Edit*.



Edit configuration

Step 28. As in *Tutorial 2*, in the FMUs section press *File* next to the Controller element, *c*. A file browser window will open and show five FMUs (if the file browser does not show the FMUs, navigate to *tutorials_4/FMUs*). Select *FMUController.fmu* and click *Open*.

1. Locate and select *FMUController.fmu*



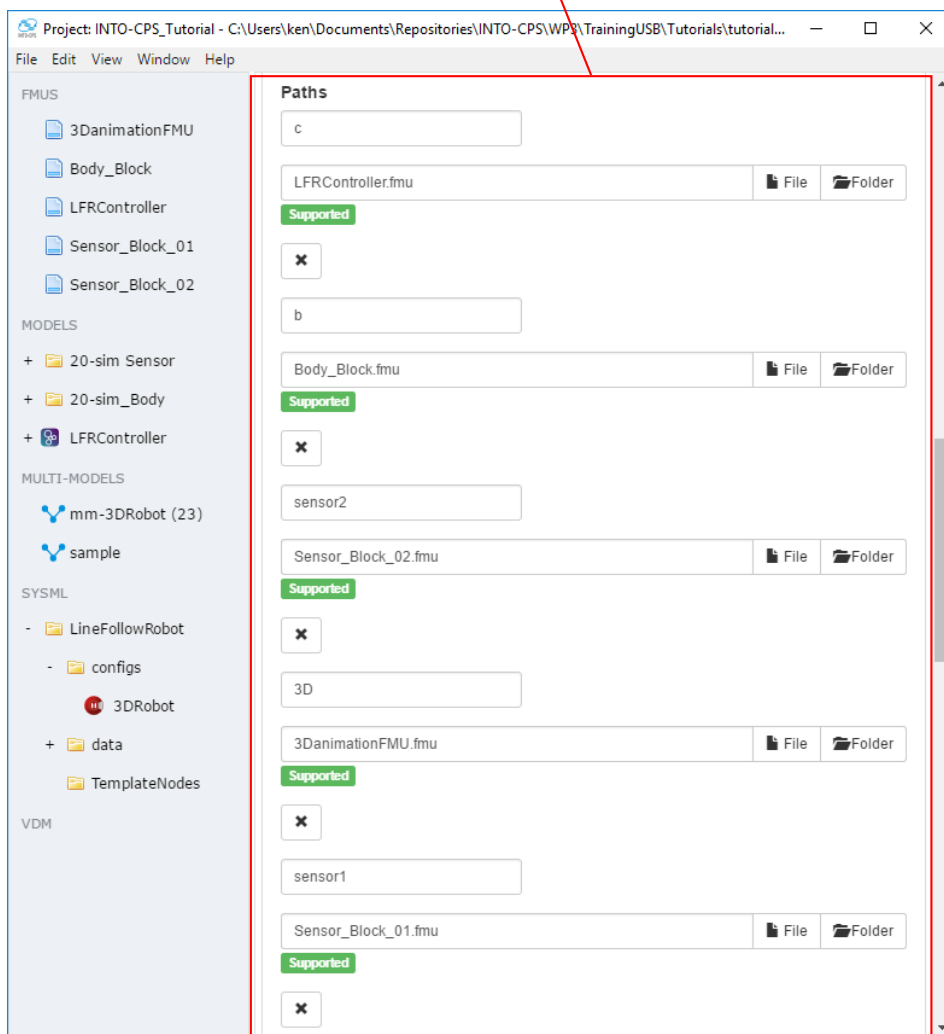
2. Click *Open*

Step 29. Repeat this for the remaining elements:

- *b* : *Body_Block.fmu*
- *3D* : *3DanimationFMU.fmu*
- *sensor1* : *Sensor_Block_01.fmu*
- *sensor2* : *Sensor_Block_02.fmu*

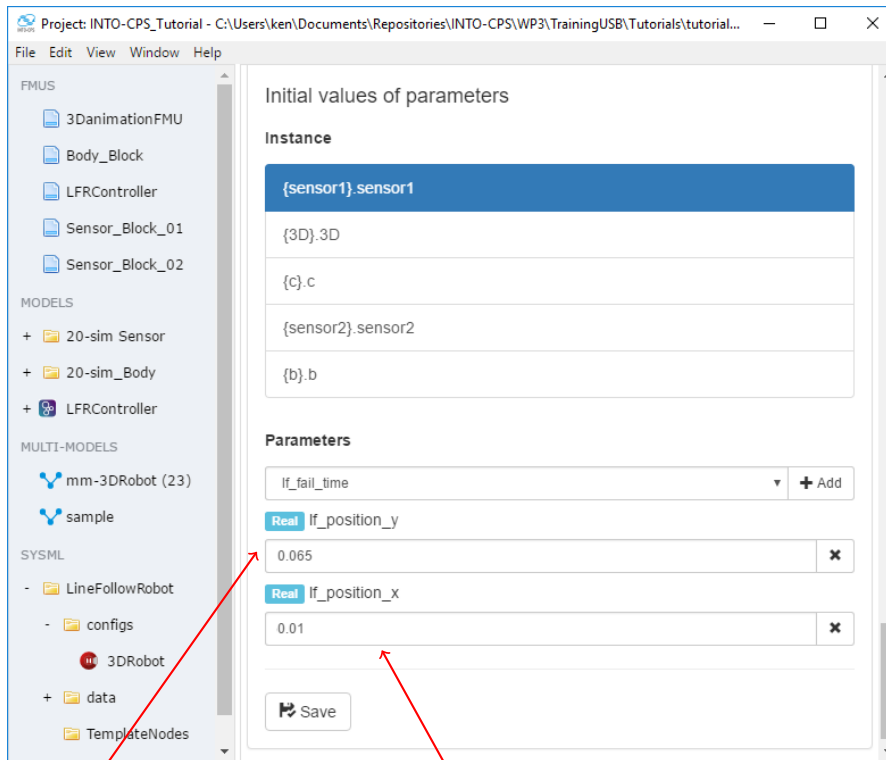
The complete set of FMUs will look like this:

FMUs added



Step 30. Scroll down to the *Initial values of parameters* section, and click $\{sensor1\}.sensor1$. In the *Parameters* section, enter the following values:

- $lf_position_y = 0.065$
- $lf_position_x = 0.01$



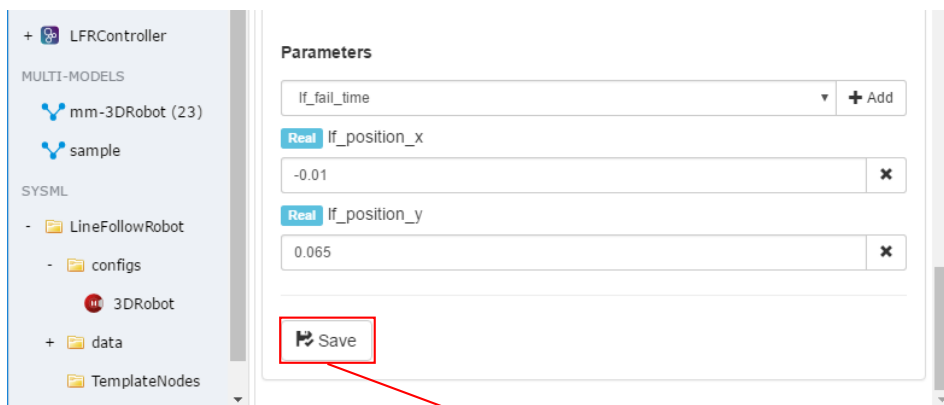
$lf_position_y$

$lf_position_x$

Step 31. Repeat the previous step for the second sensor, $\{sensor2\}.sensor2$, with the following values:

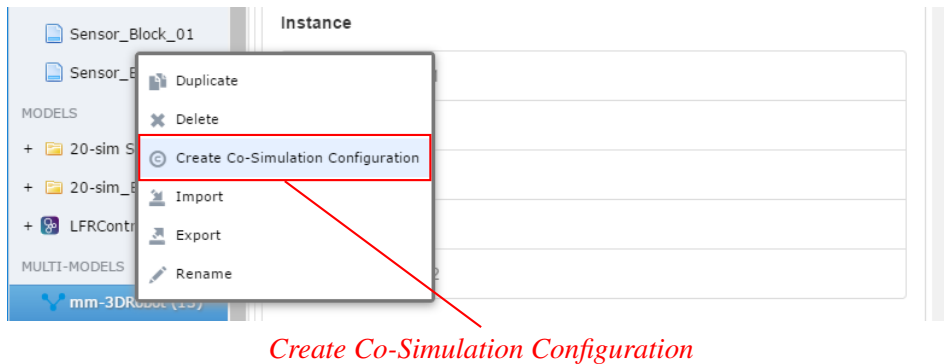
- $lf_position_x = -0.01$
- $lf_position_y = 0.065$

Step 32. *Save the Configuration.*

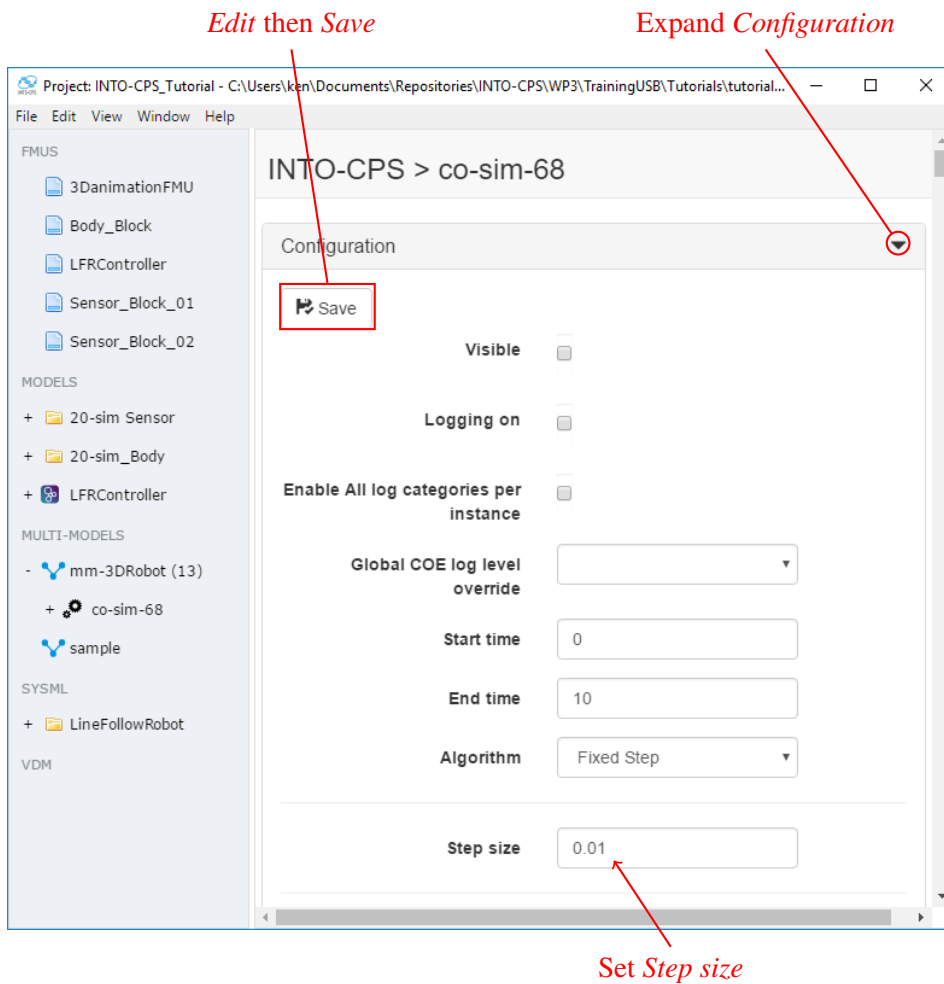


Save configuration

Step 33. Right-click on the new multi-model configuration and select *Create Co-simulation Configuration*.



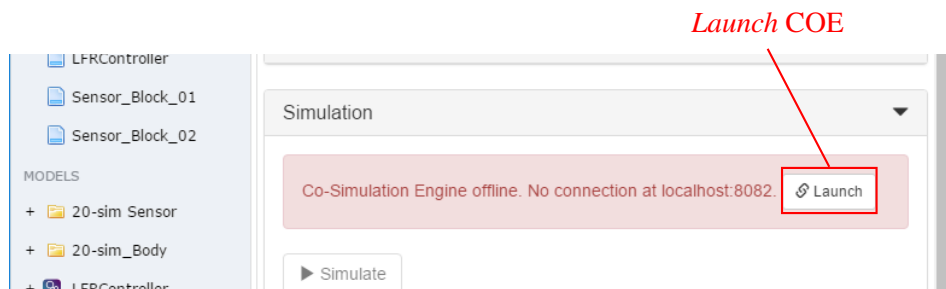
Step 34. Set the *Step size* to 0.01. Don't forget to press *Edit* then *Save*.



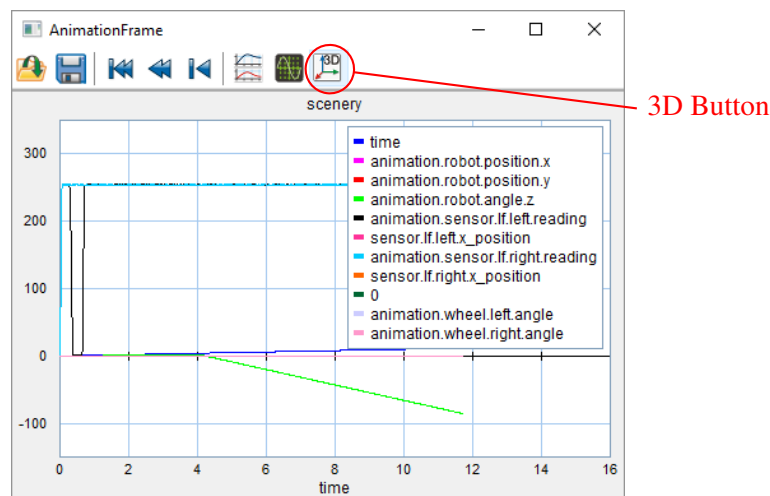
Step 35. Check *If_1_sensor_reading* from $\{sensor1\}.sensor1$ and $\{sensor2\}.sensor2$ to see the sensor values appear in the *Live Plotting*.



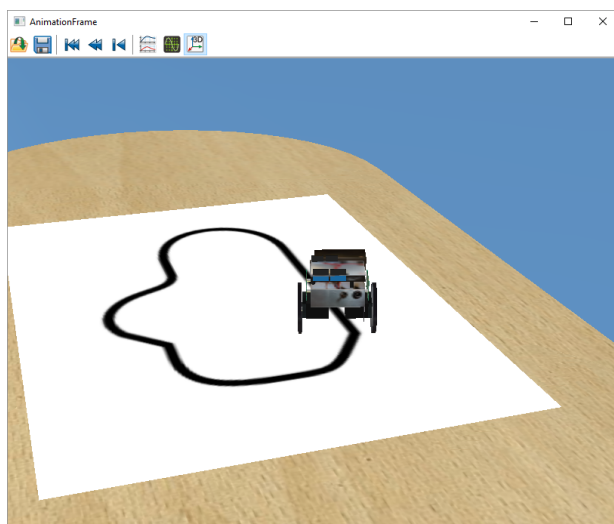
Step 36. Launch the COE if necessary (see *Tutorial 1 — First Co-simulation* for a reminder if needed).



Step 37. When the COE is running, click the *Simulate* button. If you are on Windows, the *Animation Frame* should appear. You can click the *3D* button to see the 3D visualisation of the robot.



Step 38. If everything went well, the robot should follow the line as in *Tutorial 2 — Adding FMUs*.



You can go back to *Overture* and look at the logic in `Controller.vdmrt`, and try to make some changes. Just repeat Step 20. to Step 23. to regenerate and copy the FMU, then press *Simulate*.

7 Additional Exercises

1. Run multiple co-simulations, each with a different step size (e.g., 0.01, 0.05, 0.1, 0.5, ...), and the same controller. Can you explain the differences in the results? Imagine that this controller gets deployed onto a real robot. What does the step size mean there?