Practical

Instructions

John Fitzgerald

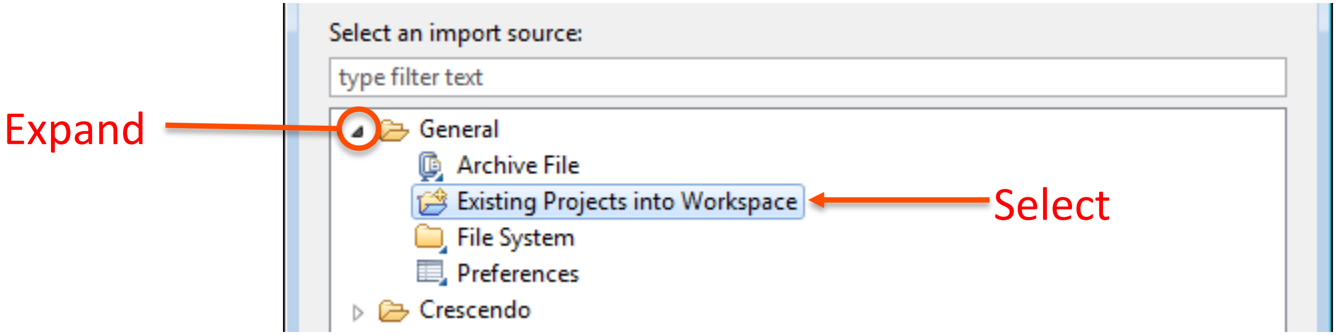
Peter Gorm Larsen

Newcastle University

AARHUS UNIVERSITY

Import the Co-model

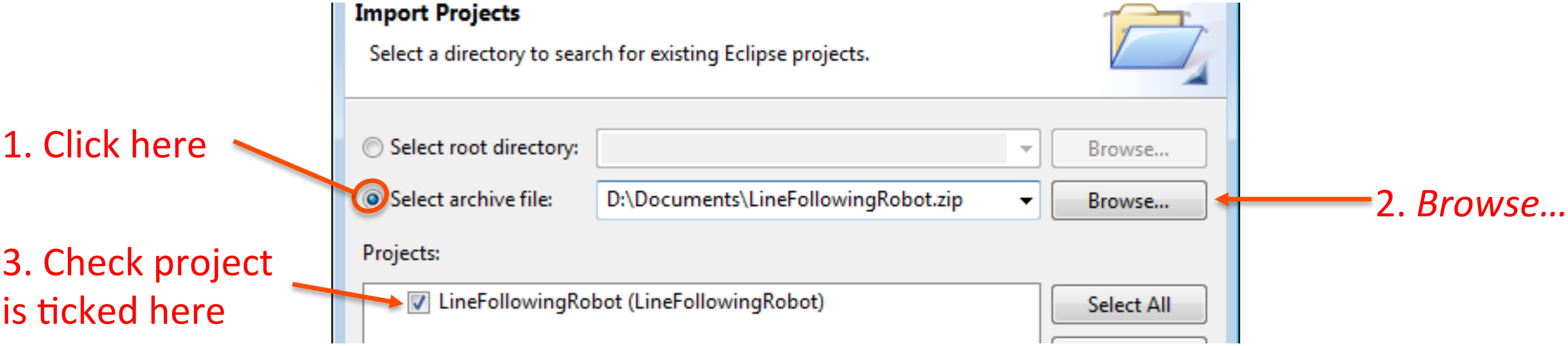
- Open Crescendo and select *File > Import…*
- Under *General* select *Existing Projects into Workspace*, then click *Next*.



Select an import source:

type filter text

- Browse for the *Robot\LineFollowingRobot.zip*:



Import Projects

Select a directory to search for existing Eclipse projects.

○ Select root directory:

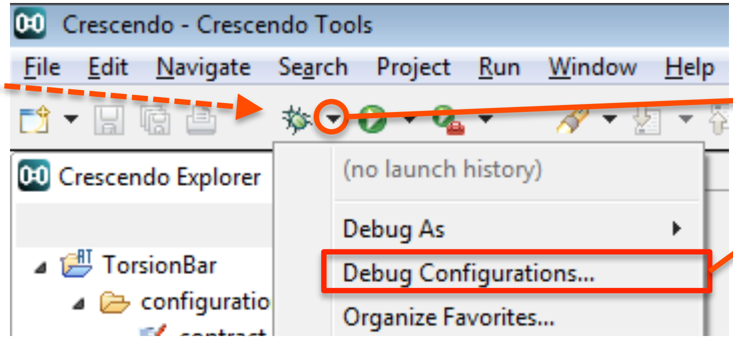
● Select archive file: D:\Documents\LineFollowingRobot.zip

Projects:

☑ LineFollowingRobot (LineFollowingRobot)

Run the Co-simulation

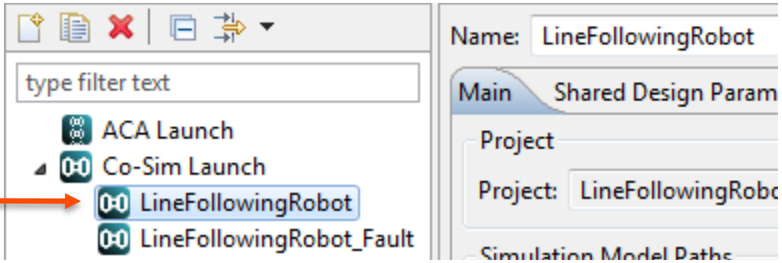
- Open *Debug Configurations…*



Debug

1. Debug menu
2. Debug Configurations…

- Crescendo will scan the new co-model and will pick up two launch configurations that are already included:



Name: LineFollowingRobot

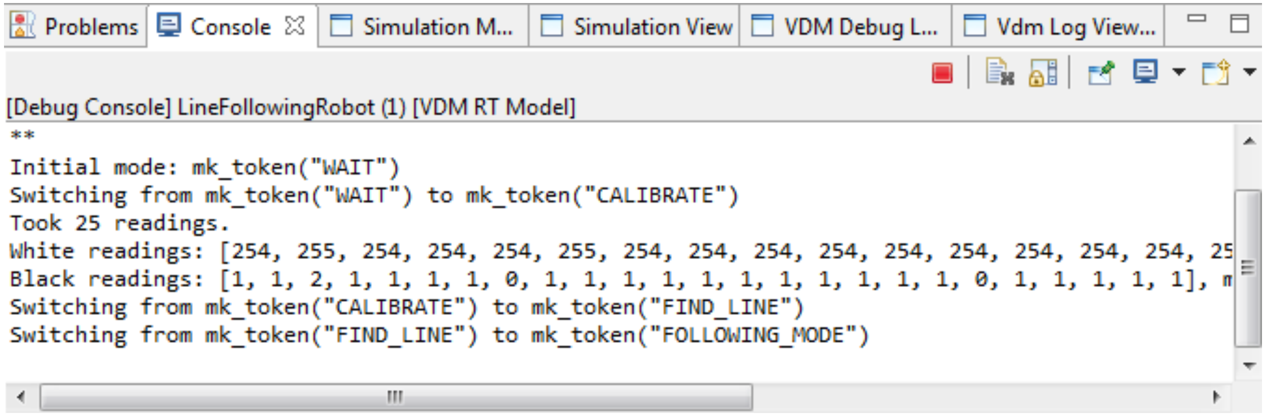
Project: LineFollowingRobc

Select

- Select *LineFollowingRobot* and hit *Debug*

Observing the Co-simulation (1)

- The robot drove off the table! This is because the control logic has been removed from the *FollowMode* class.
- The controller delegates the control of the robot wheels to the mode objects (calibrating sensors, finding and following the line). Mode changes appear in the console:



[Debug Console] LineFollowingRobot (1) [VDM RT Model]

```
**
Initial mode: mk_token("WAIT")
Switching from mk_token("WAIT") to mk_token("CALIBRATE")
Took 25 readings.
White readings: [254, 255, 254, 254, 254, 255, 254, 254, 254, 254, 254, 254, 254, 254, 254, 25
Black readings: [1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1], m
Switching from mk_token("CALIBRATE") to mk_token("FIND_LINE")
Switching from mk_token("FIND_LINE") to mk_token("FOLLOWING_MODE")
```

- Each mode is written as a class. These classes are then instantiated as mode objects.

Fix the *FollowMode* Class

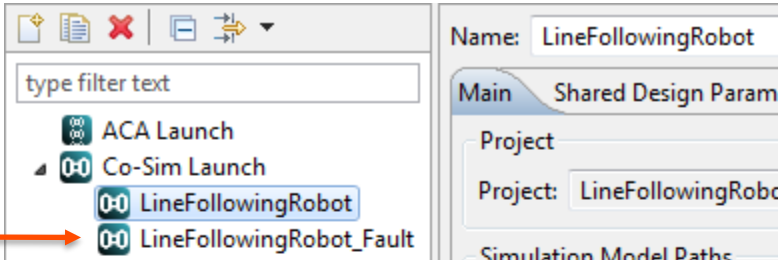
- Open *model_de\controller\modes\FollowMode.vdmrt*
- Change the body of the Step() operation to the following:

```
-- control loop
public Step: () ==> [AbstractModalController`Mode]
Step() == (
    -- black to the right, go right
    if con.GetLeftBW().IsWhite() and con.GetRightBW().IsBlack()
    then con.TurnRight();

    -- black to the left, go left
    if con.GetLeftBW().IsBlack() and con.GetRightBW().IsWhite()
    then con.TurnLeft();

    -- no internal mode change
    return nil;
);
```

- Rerun the co-simulation. The robot should follow the line!

Activate the Fault

- Open *Debug Configurations…* again
- This time select *LineFollowingRobot_Fault* and click *Debug*



Name: LineFollowingRobot

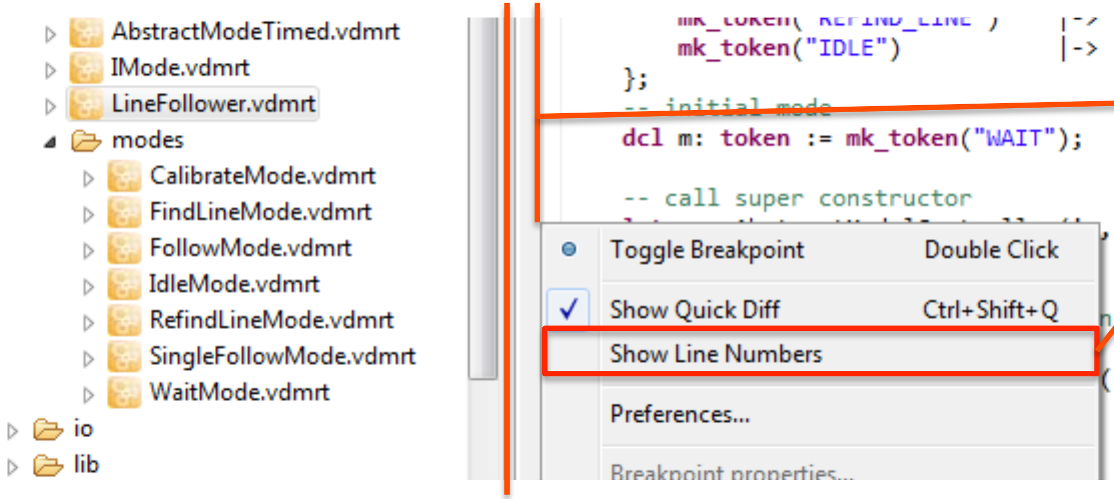
Project: LineFollowingRobo

Select

- In this co-simulation configuration, the left sensor will fail and become stuck. This means it will always read black
- Note the behaviour of the robot (a *failure* of the system to follow the line due to a *failed* component)

Enable the Mode Change

- The controller is already able to change mode when the sensor fails, but the statement is commented out
- Open *model_de\controller\LineFollower.vdmrt*
- Enable line numbers by right-clicking in the “gutter”



```
    mk_token("IDLE")            |->
};
-- initial mode
dcl m: token := mk_token("WAIT");

-- call super constructor
```

1. Right-click somewhere here (in the “gutter”)

2. Left-click here

- Uncomment lines 45—46 of *LineFollower.vdmrt*
  - delete the -- (single line comment character); or
  - select the lines and press *Ctrl+F7*

Adding a Single-sensor Mode

- Re-run the *LineFollowingRobot_Fault* co-simulation.
- The robot should stop after the failure occurs. The robot now “fails safe” when the fault is detected
- It is possible however to offer a degraded mode of behaviour by using the remaining working sensor
- Your final task is to implement the Step() operation in *model_de\controller\modes\SingleFollowMode.vdmrt* to follow the line using the right sensor only
- A partial solution to this task is provided in the .zip file
- Paste the contents of *Robot\SingleFollowMode.vdmrt* into *model_de\controller\modes\SingleFollowMode.vdmrt* (replacing all the existing contents)

Understanding the Partial Solution

- Look at the class you just pasted in and observe the following:
  - Line following with a single sensor can be done by detecting the edge of the line (changes from black to white and back), then switching direction
  - This class tracks the previous colour and direction using instance variables and custom types
  - It is easy to lose the line in this mode, so the class also counts how many controller cycles pass with only white being detected (indicating the line has been lost). This class then switches to a mode called *RefindLine*, by returning its id from *Step()*, which sweeps back and forth until black is seen again, then the mode switches back to *SingleFollowMode*.

Completing the Task

- If you run the *LineFollowingRobot_Fault* co-simulation again, you will see that the robot follows the line, though inefficiently.
  - because it detects the edge by timing-out while over white
- To complete this task and improve the performance, you need to insert an if-statement at line 46 of *SingleFollowMode.vdmrt* to detect the edge of the line and change direction.
  - a comment is included to help (see lines 47+48)
- Note that you might need to turn slower than *con.TurnLeft()* and *con.TurnRight()* provide
  - this can be done by calling con.Turn(x,y), where x and y are between -1 and 1 (full speed back- and forwards, respectively)

Solution

- The following if-statement should allow the SingleFollow mode to work:

```
-- moved off line, change direction
if colour = <WHITE> and prev_colour = <BLACK> then
    if prev_dir = <LEFT> then (
        con.Turn(0.3, 0.0);
        prev_colour := <WHITE>;
        prev_dir := <RIGHT>
    )
    else (
        con.Turn(0.0, 0.3);
        prev_colour := <WHITE>;
        prev_dir := <LEFT>
    );
```