Practical

Instructions

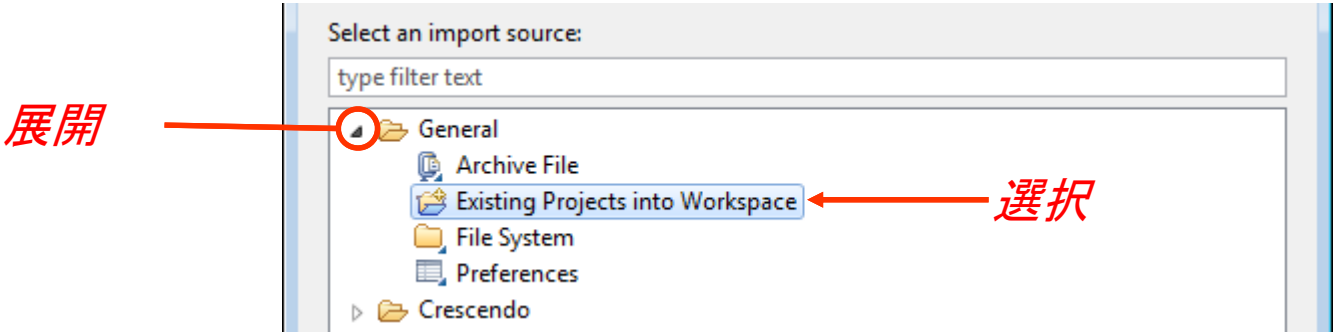
John Fitzgerald

Peter Gorm Larsen

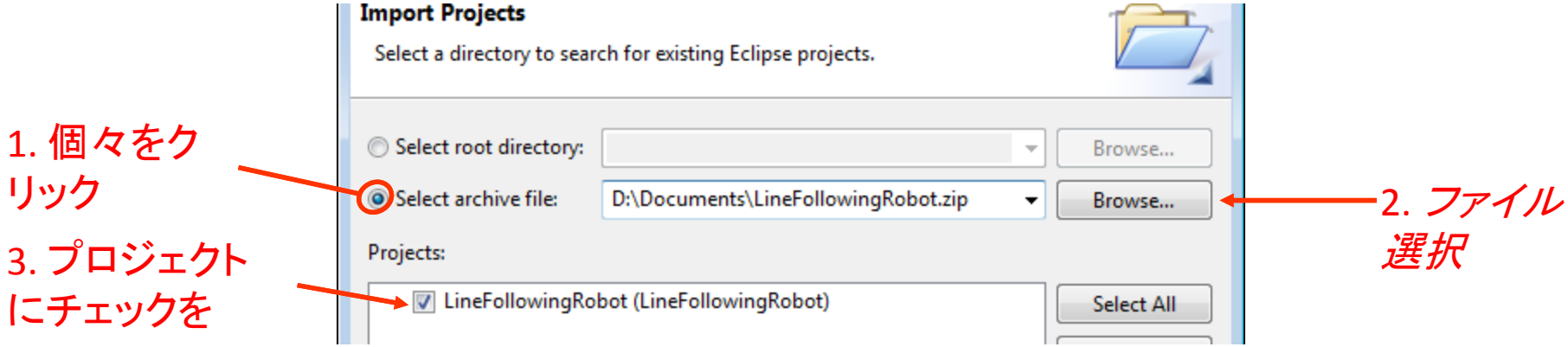
AARHUS
UNIVERSITY

Co-modelのインポート

- Crescendoを立ち上げてメニューから *File > Import*
- *General* の中から *Existing Projects into Workspace* を選択して *Next* を押す

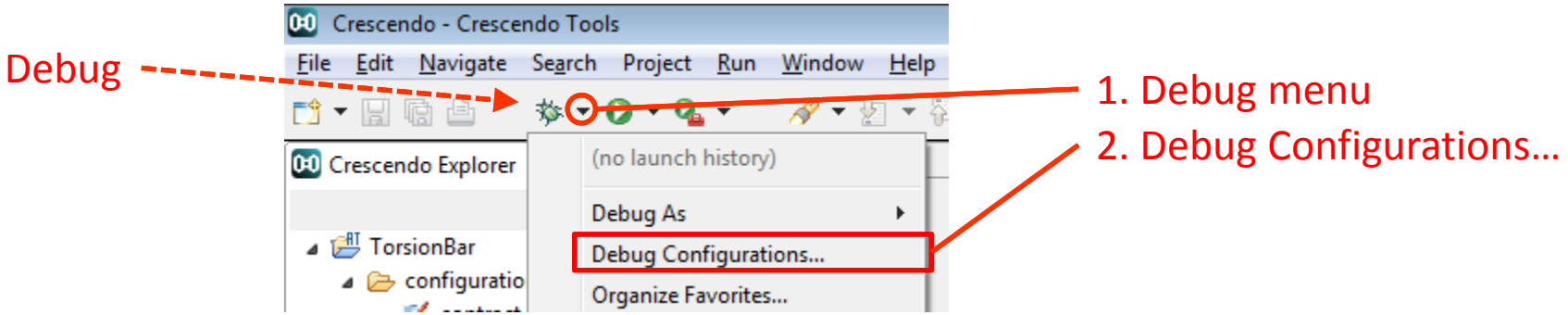


- Browse for the *Robot¥LineFollowingRobot.zip*:

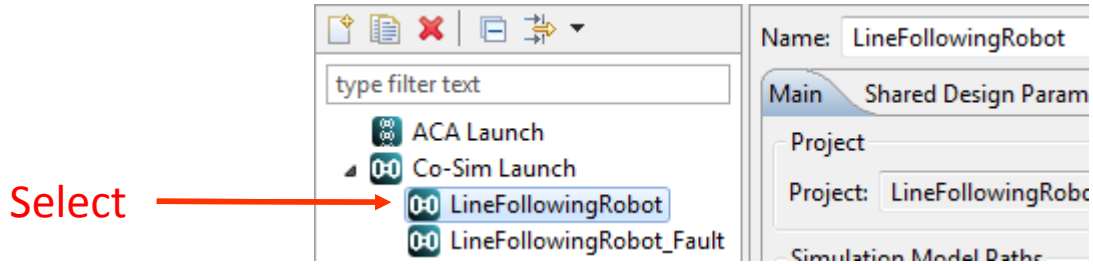


Co-simulationの起動

- *Debug Configurations* を開く



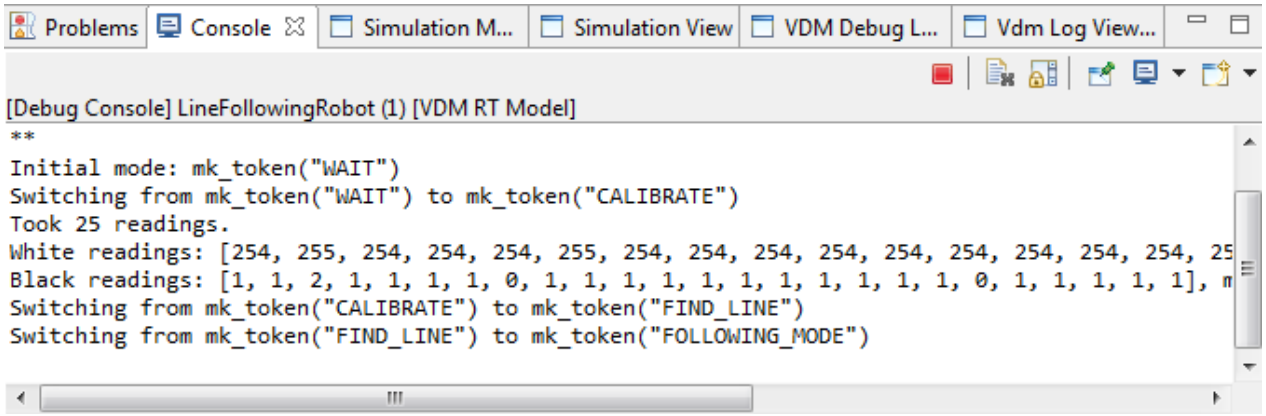
- Crescendo は読み込んだco-modelから，定義された起動設定（launch configurations）を読んでおいてくれる：



- *LineFollowingRobot* を選んで *Debug* を押す

Co-simulationの観察 (1)

- ロボットはテーブルから走り落ちる！これは制御ロジックが *FollowMode* クラスから削除されているため
- コントローラーは，ロボットの車輪制御をモードオブジェクトに委譲している（センサーを読んで線を発見，それに沿って動く）．モードはコンソールログに出力される:



- 各モードはクラスとして定義され，オブジェクトとしてインスタンス化される

FollowMode クラスの修正

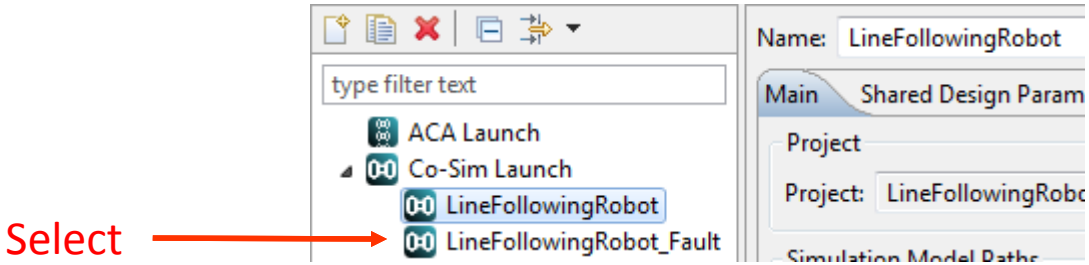
- *model_de¥controller¥modes¥FollowMode.vdmrt* を開く
- Step() 操作の本体（振る舞い定義）を以下に変える:

```
-- 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;
);
```

- co-simulationを走らせるとロボットは線に沿って走る！

障害を有効にする

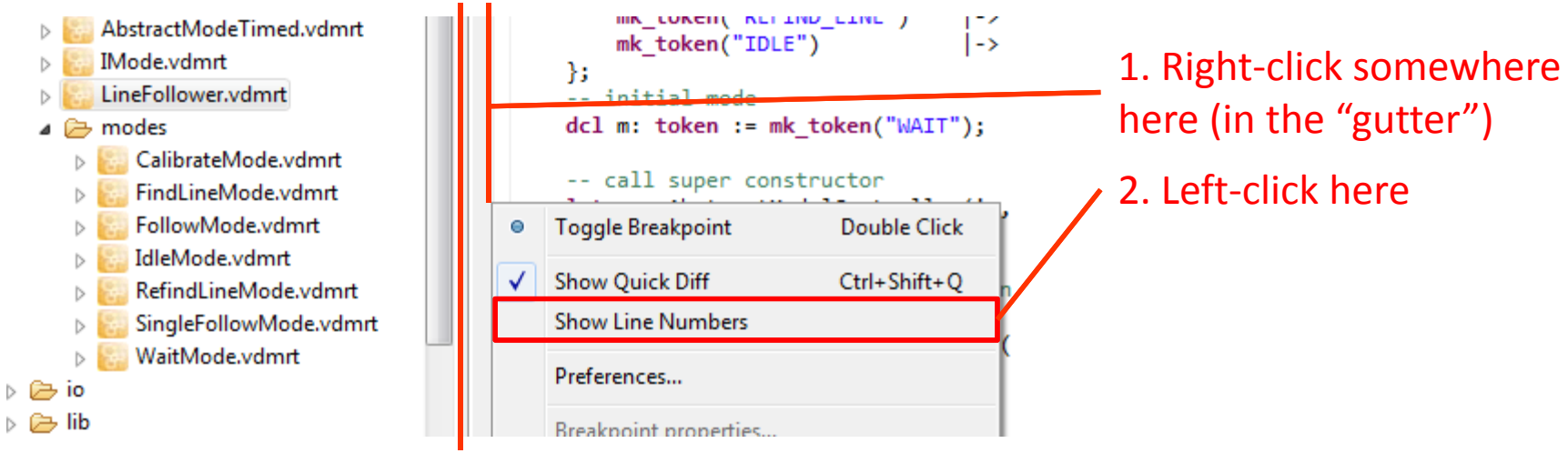
- 再度 *Debug Configurations* を開く
- 今度は *LineFollowingRobot_Fault* を選んで *Debug* をクリック



- この co-simulation 設定では，左側のセンサーが壊れて固まってしまう（常に黒を出力）
- ロボットの振る舞いに着目（障害がある部品により，線に沿って走るというシステム機能の故障）

Enable the Mode Change

- コントローラーはセンサー障害時にモードを変えるようになっているが，コメントアウトされている
- *model_de¥controller¥LineFollower.vdmrt* を開く
- 脇の溝を右クリックして行数を表示



- *LineFollower.vdmrt* の45-46行目のコメントを外す
  - -- （ハイフン2つ）を消す
  - か，行を選んで *Ctrl+F7*

Single-sensor Modeを追加

- 再度 *LineFollowingRobot_Fault* co-simulationを実行
- 今度はロボットが障害発生時に止まってしまう(フェールセーフになっている).
- しかし，片方のセンサーだけでも退縮して(デグレードして)振る舞い続けることが可能
- 最後の演習は下記ファイルの Step() 操作を設計して，右のセンサーだけでも走り続けられるようにすること
  model_de¥controller¥modes¥SingleFollowMode.vdmrt
- 部分的な解は.zipファイル内にある:
- *Robot¥SingleFollowMode.vdmrt* の中身で，*model_de¥controller¥modes¥SingleFollowMode.vdmrt* の既存部分をすべて置き換える

部分的な解の理解

- この貼り付けたクラスを見てみると:
  - 単一センサーだけで線に沿って進むことは，線の縁（黒から白へ，白から黒への変化）を検出して方向を変えることで実現できる
  - このクラスは，インスタンス変数と専用の型により以前の色と方向を覚えておいている
  - 線を見失うことも簡単に起きてしまうので，白のみを検出したパスがどれだけ続くかを数えて見失ったことに気づくようにしている．その場合 *RefindLine* というモードのIDをreturnしてそのモードに変更し，再度黒が検出されるまで円を描く．黒が見つかったらモードは *SingleFollowMode* に戻す

完成させよう！

- 再度 *LineFollowingRobot_Fault* co-simulationを実行すると，ロボットは線に沿って進むがとても非効率だとわかる
  - 白の上でタイムアウトして縁を見つけるため
- 性能を向上するためには，*SingleFollowMode.vdmrt* の46行目にif文を挿入し，線の縁を検出して方向を変える必要がある
  - コメントが参考に添えてある（47-48行目）
- ただし，ロボット制御の操作として，*con.TurnLeft()* や *con.TurnRight()* よりもゆっくり回る必要がある
  - このため，con.Turn(x,y) を用いる：
    xとyは左右のエンジン出力（-1から1の間の値，-1は全力後進，1は全力前進）