

HybridSynchAADL

Tutorial

Outline

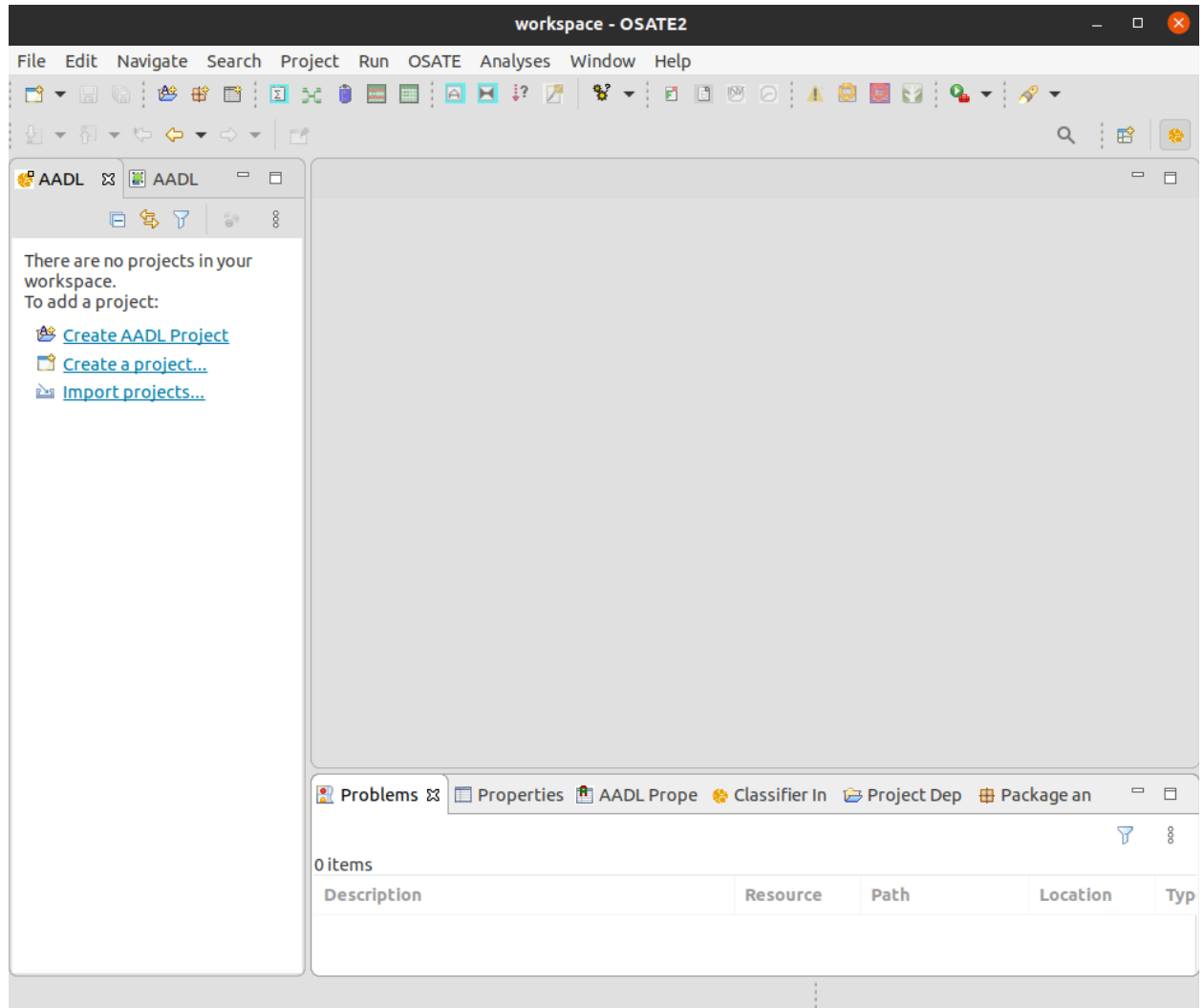
1. Basic OSATE
2. Creating Property Specification Files (PSPC)
3. HybridSynchAADL Constraints Checker
4. Maude Code Generation
5. Formal Analysis

Outline

1. Basic OSATE
2. Creating Property Specification Files (PSPC)
3. HybridSynchAADL Constraints Checker
4. Maude Code Generation
5. Formal Analysis

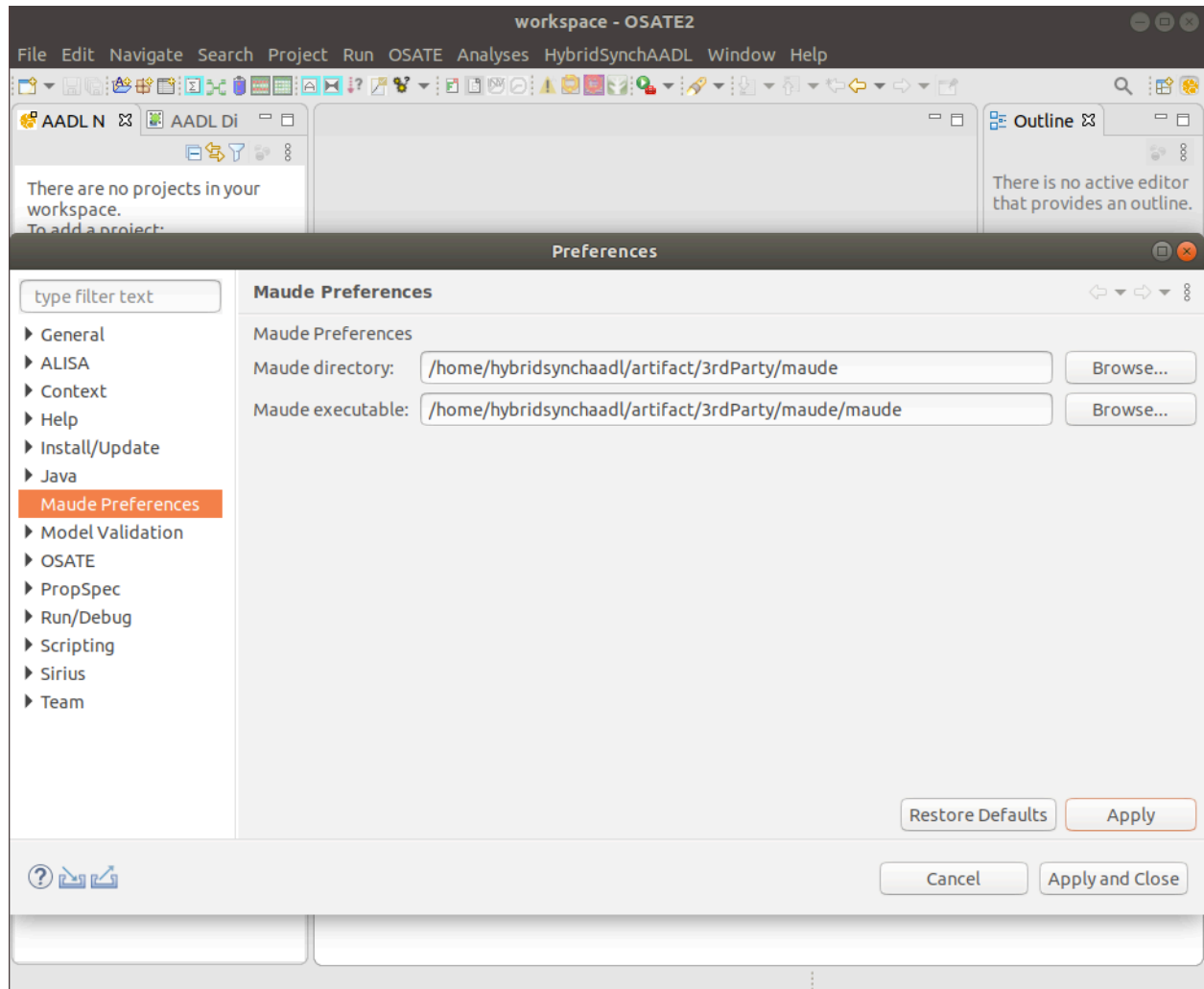
Running OSATE

- You will see this window when you execute OSATE.



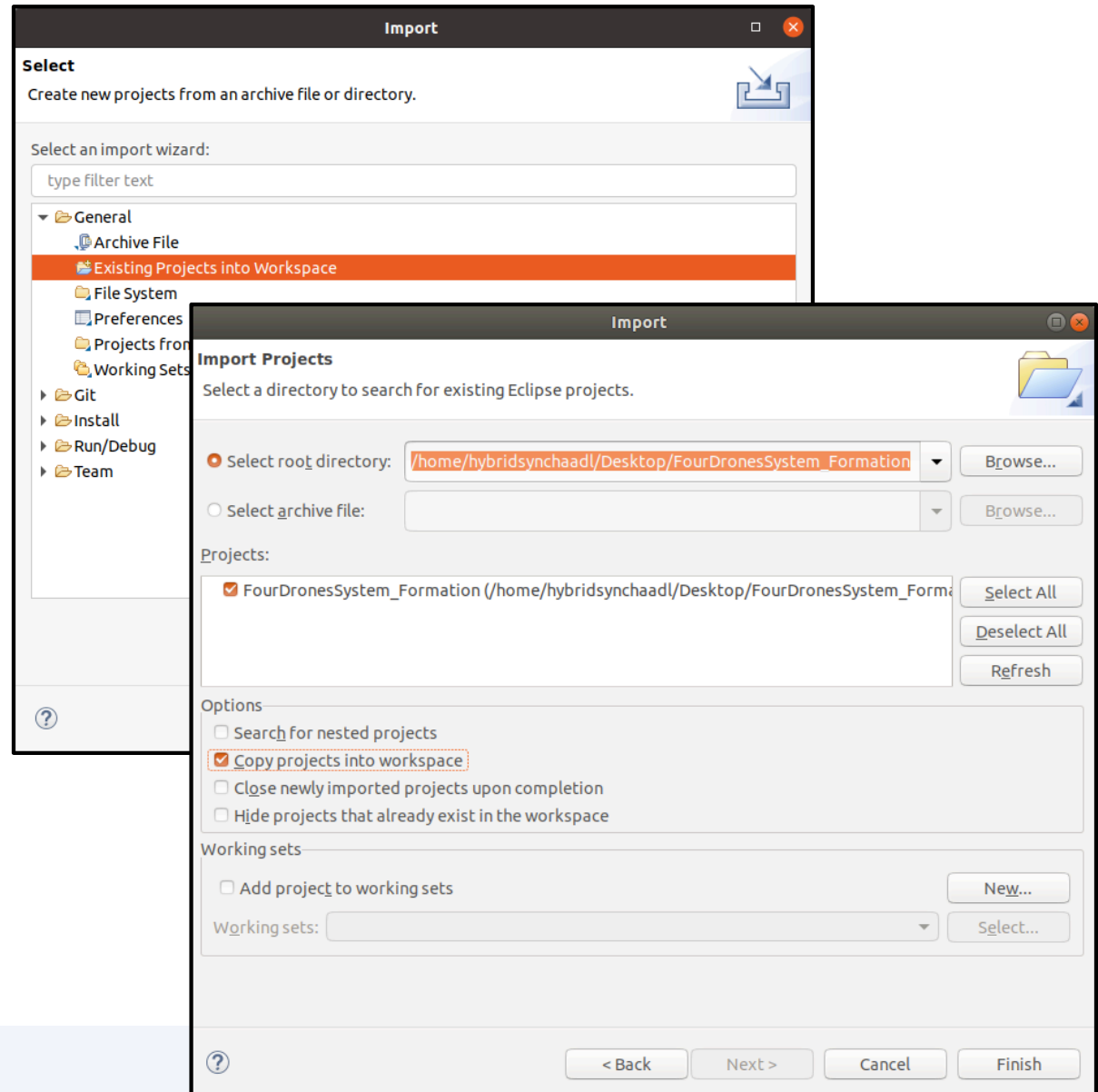
Maude Preferences

- Before importing an example project, set the proper Maude preferences.
- Open Preferences menu by clicking Menu → Window → Preferences.
- Set Maude directory and executable file location.



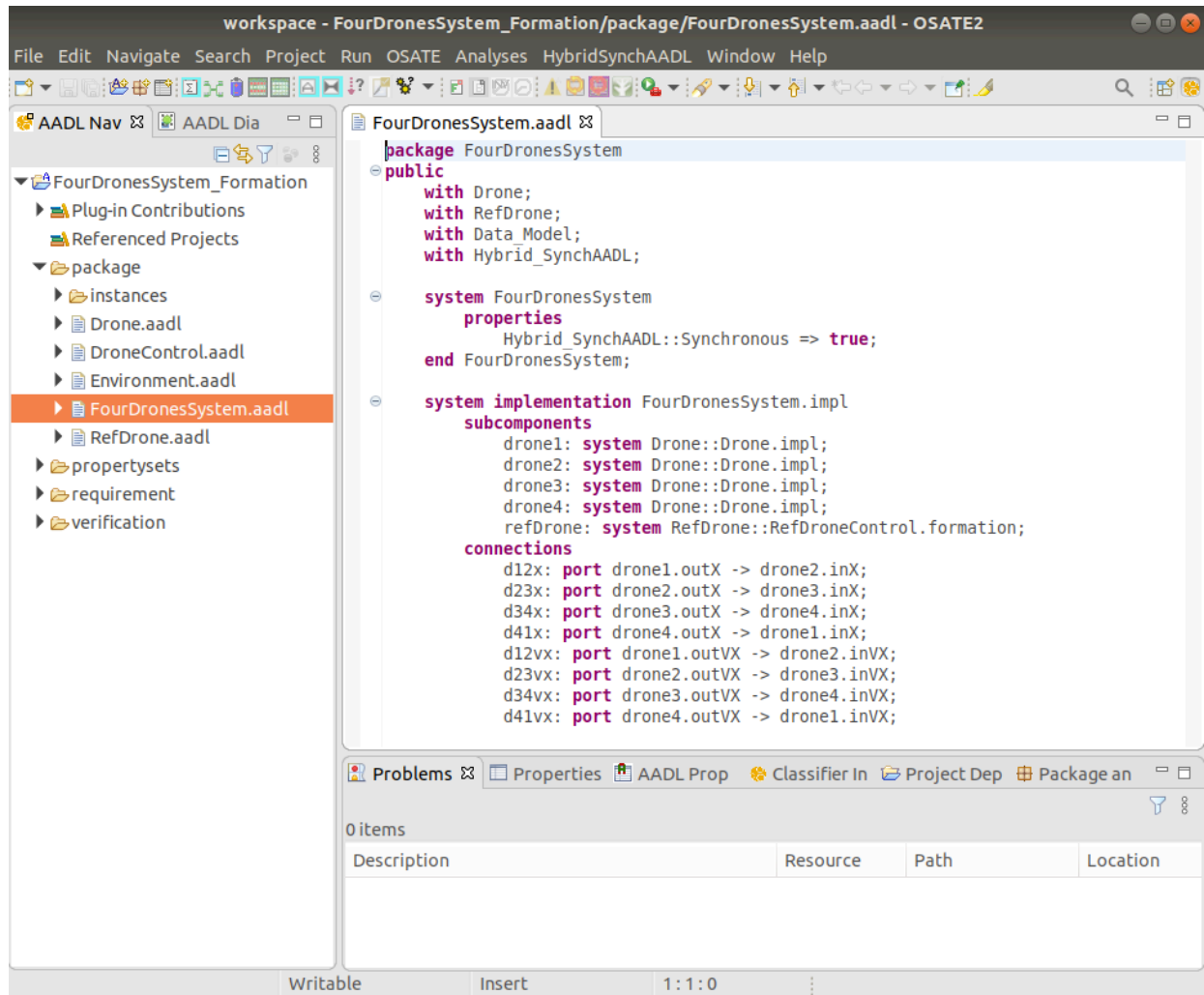
OSATE - Importing an Example

- We start with a simple example, namely, `FourDronesSystem_Formation` in the directory `models/hybridsynchaad1`.
- To import the example, choose
 - Menu → File → Import → General → Existing Projects into Workspace.



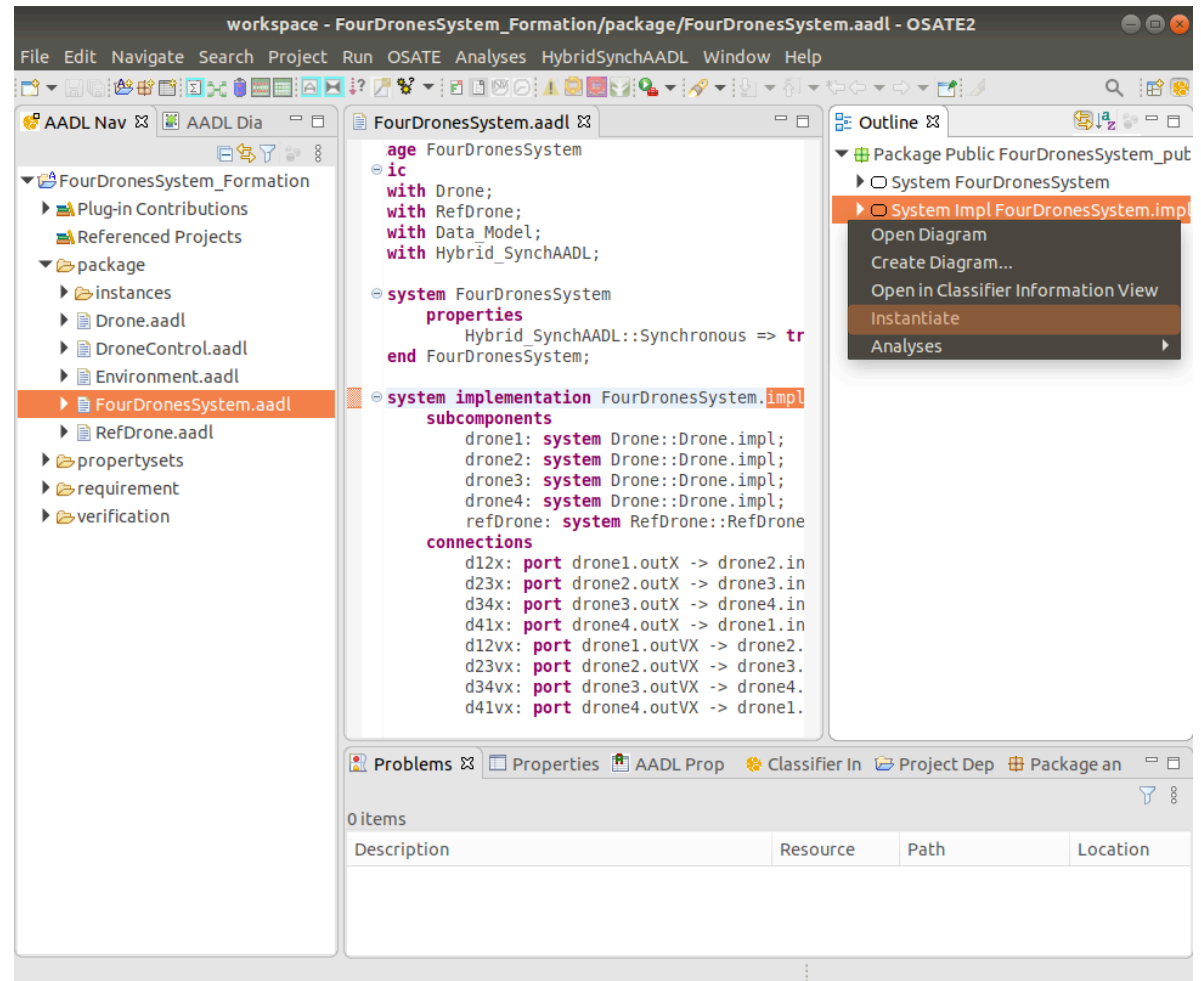
FourDronesSystem – Text

- FourDroneSystem.aadl contains the top-level system component.



Instance Model

- Open the Outline view by clicking Menu → Window → Show view → Outline.
- Create an instance model from a system implementation as follows:
 - Right click on System Impl FourDronesSystem.impl and choose Instantiate.

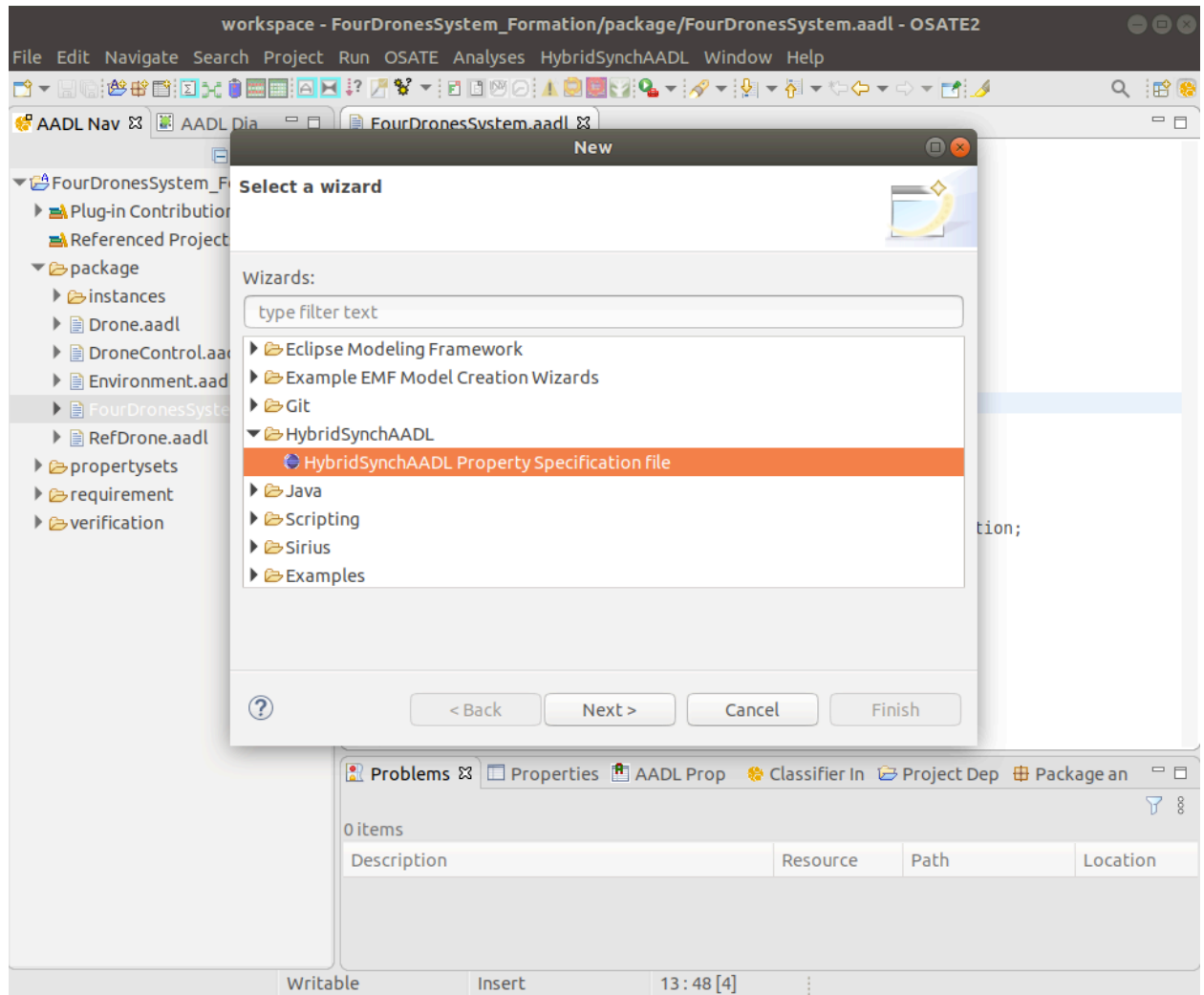


Outline

1. Basic OSATE
- 2. Creating Property Specification Files (PSPC)**
3. HybridSynchAADL Constraints Checker
4. Maude Code Generation
5. Formal Analysis

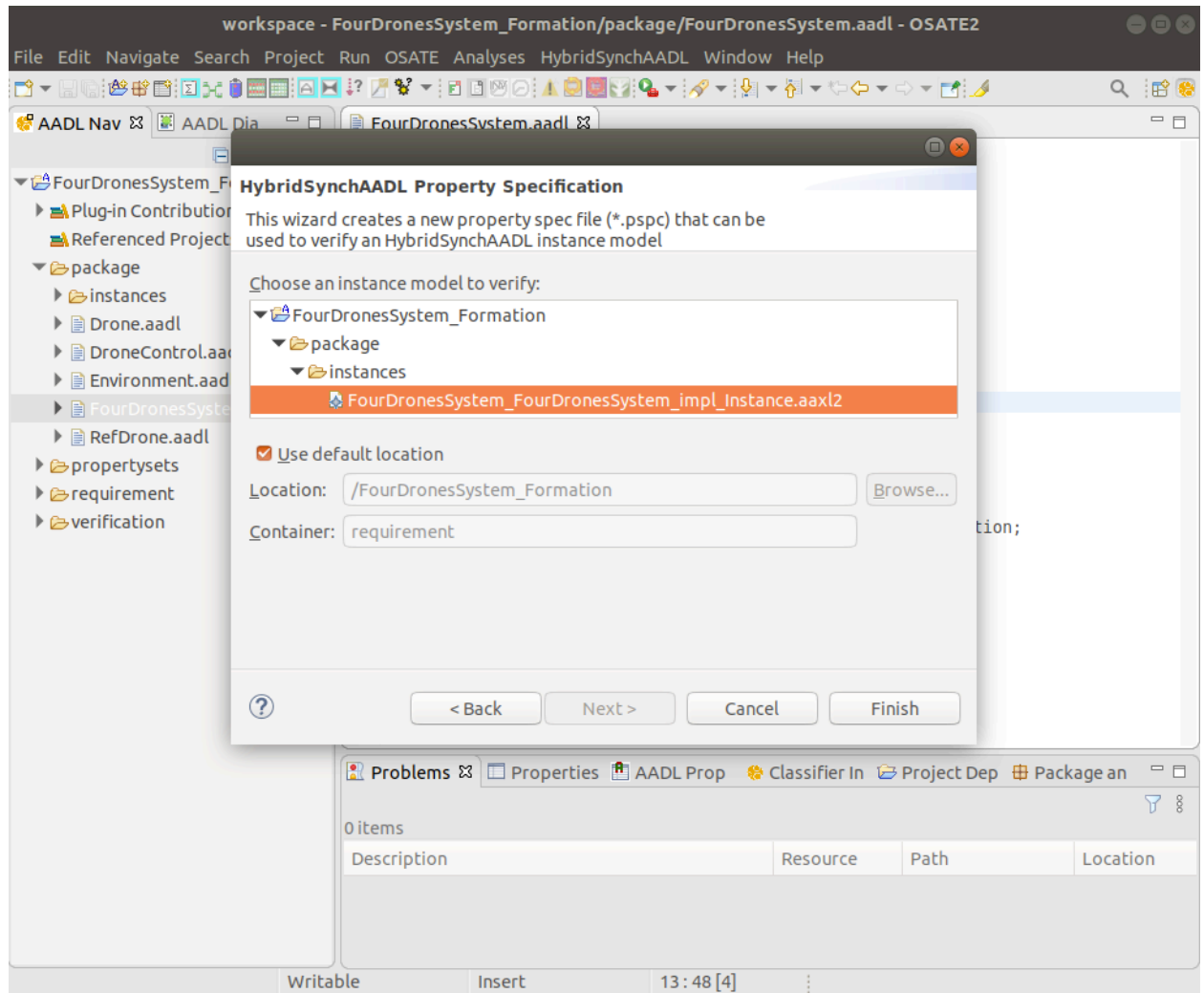
Creating PSPC Files

- To create a PSPC file, choose
 - Menu → File → New → Other → HybridSynchAADL → HybridSynchAADL Property Specification file.



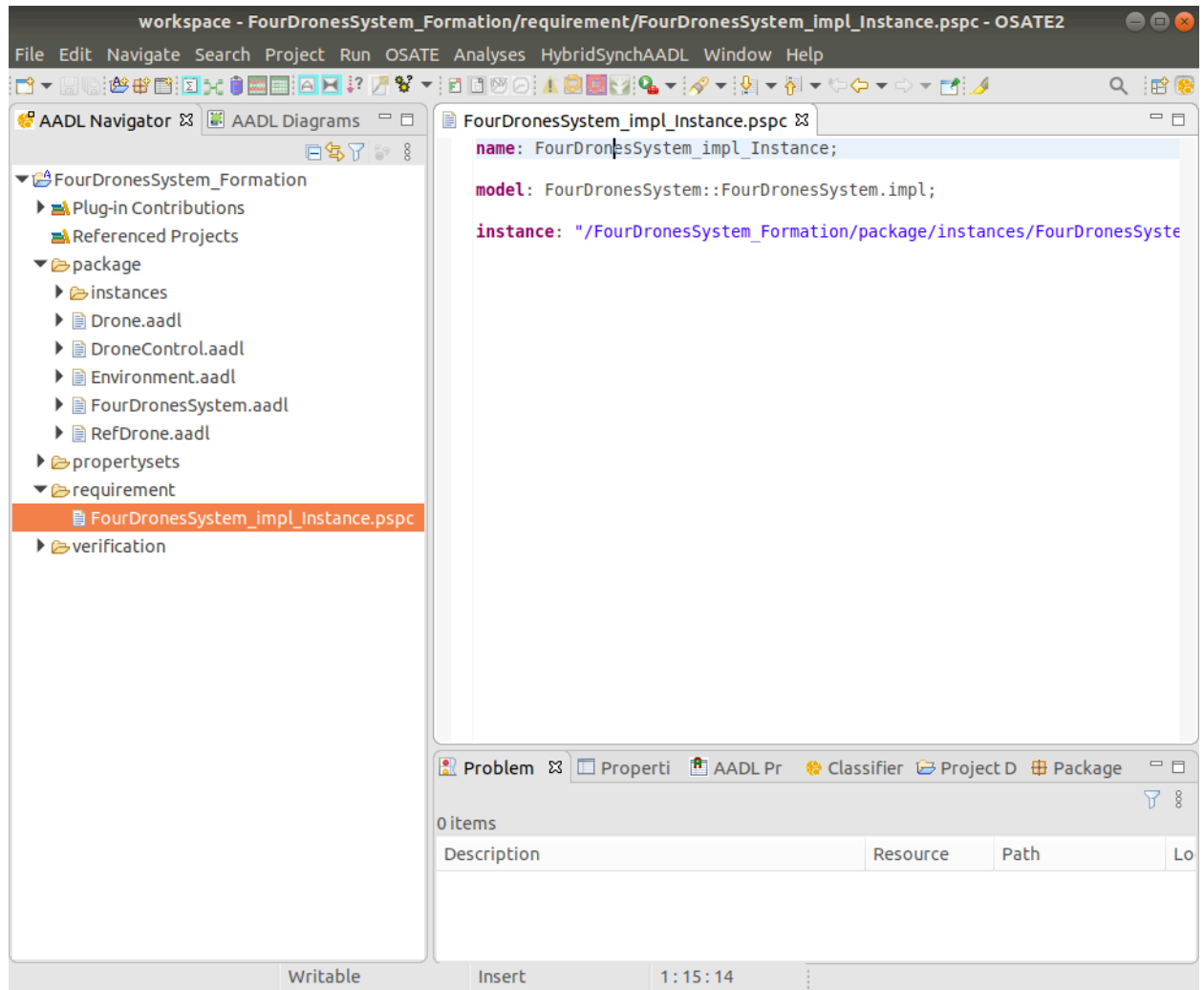
Creating PSPC Files

- Any valid AADL instance model can be chosen in the wizard.
- Choose the instance model we have created in the previous slides.



Creating PSPC Files

- This screen shows the generated (empty) PSPC file.
- There are sample PSPC file in this project

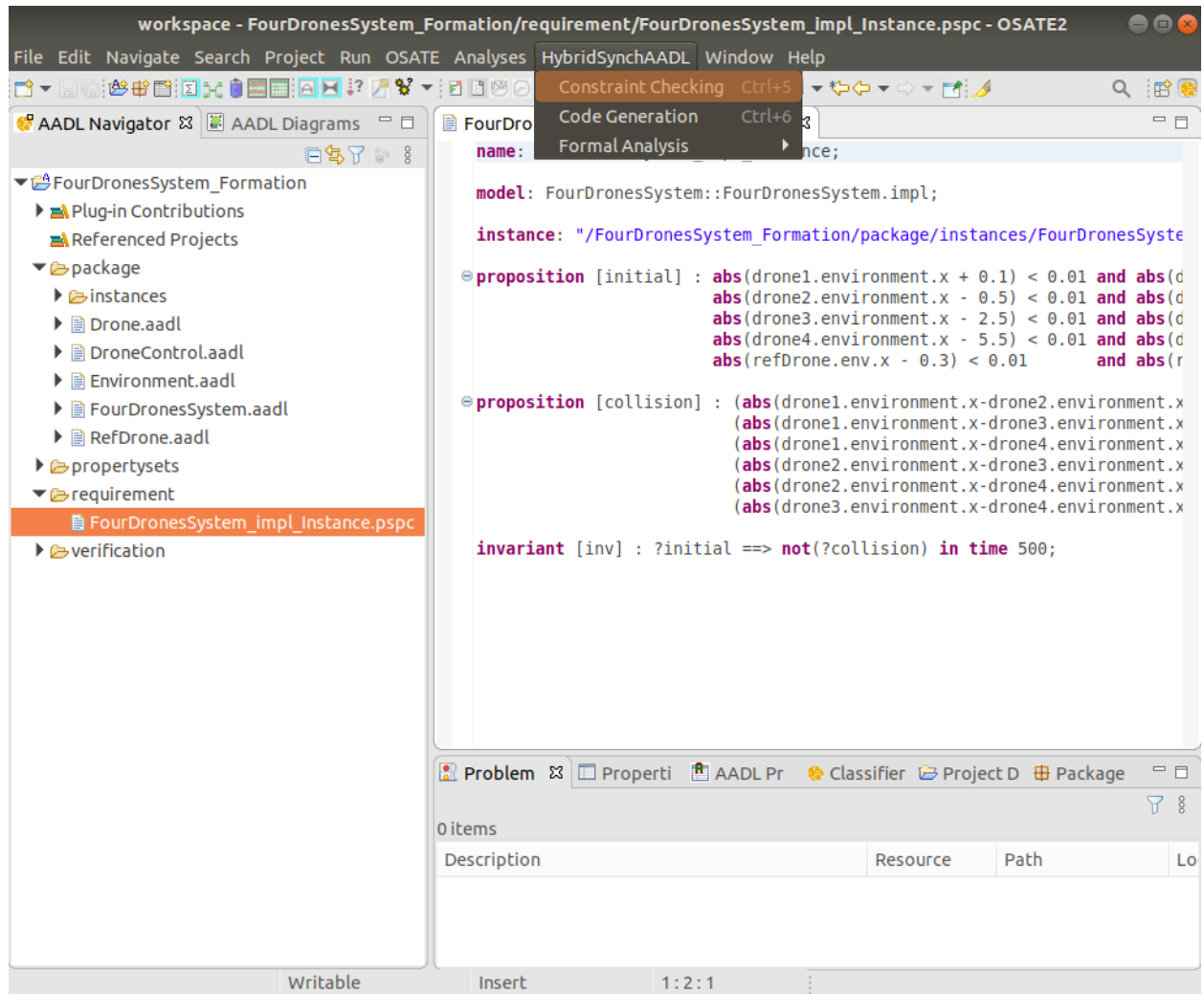


Outline

1. Basic OSATE
2. Creating Property Specification Files (PSPC)
- 3. HybridSynchAADL Constraints Checker**
4. Maude Code Generation
5. Formal Analysis

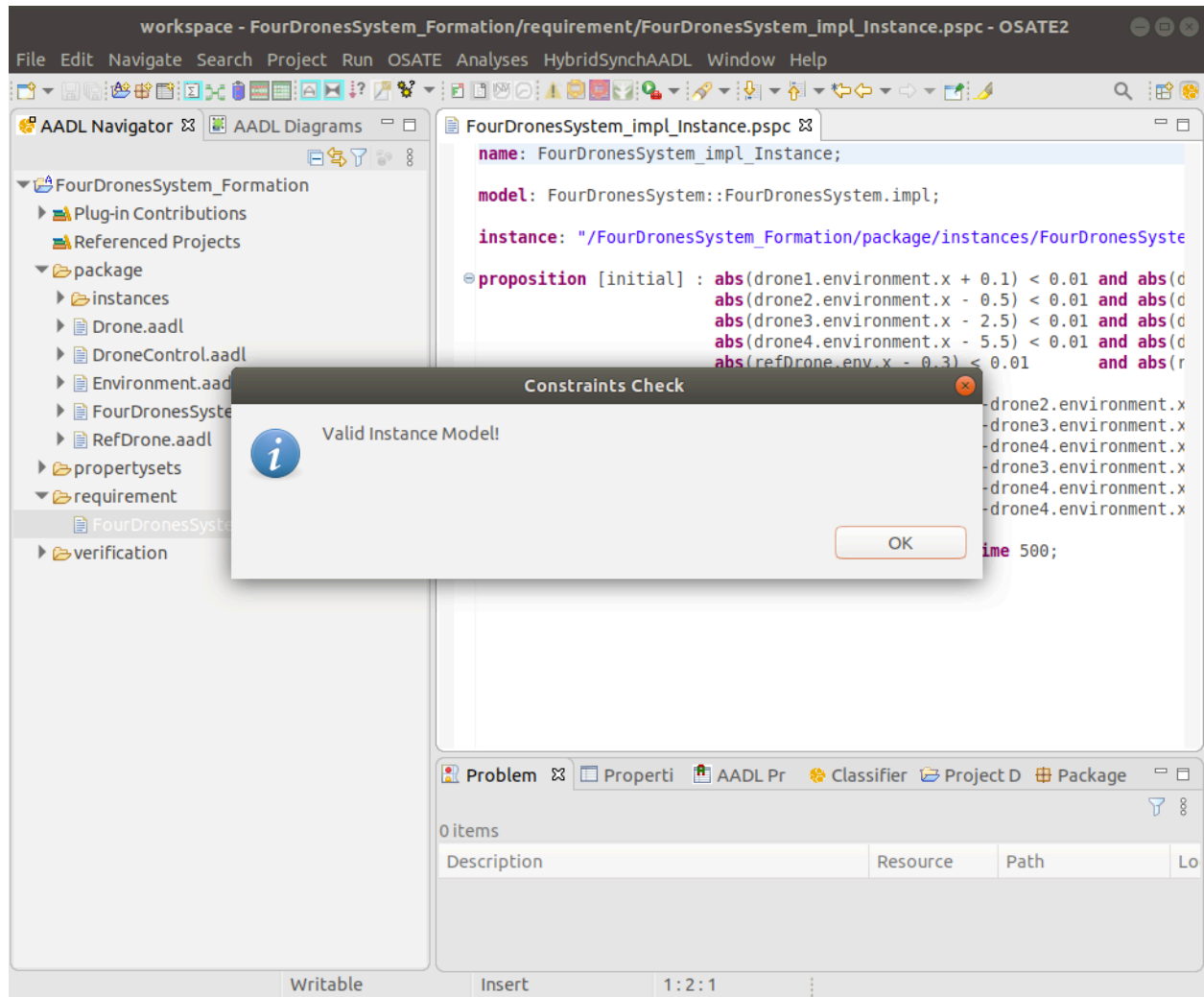
Checking HybridSynchAADL Constraints

- There are three menu items in HybridSynchAADL: Constraints Check, Code Generation, and Formal Analysis.
- Click Constraints Check to perform constraints checking.



Checking HybridSynchAADL Constraints

- When the model has no constraints error, the tool notifies that the model is valid.



Constraints Check – Erroneous Model

- What if some HybridSynchAADL constraints is not satisfied?
- Let us add an invalid initial value to data component and see what happened.
 - by changing the property value param => SomethingWrong.

The screenshot shows the OSATE2 workspace for a project named 'FourDronesSystem_Formation'. The left pane displays the AADL Navigator with a tree structure. The 'FourDronesSystem.aadl' file is selected. The right pane shows the AADL code for 'FourDronesSystem_impl_Instance.pspc'. The code defines several ports (r1x, r2x, r3x, r4x, r1vx, r2vx, r3vx, r4vx, r1y, r2y, r3y, r4y, r1vy, r2vy, r3vy, r4vy) and their connections to drone components. Below the code, the 'properties' section is visible, showing three initial value constraints. The first constraint, 'Data_Model::Initial_Value => ("SomethingWrong") applies to drone1.environment.x, drone2.environment.x, drone3.environment.x, drone4.environment.x;', is highlighted with a red arrow. The second and third constraints are 'Data_Model::Initial_Value => ("0") applies to drone1.environment.dotx, drone2.environment.dotx, drone3.environment.dotx, drone4.environment.dotx;' and 'Data_Model::Initial_Value => ("0") applies to drone1.environment.dotdotx, drone2.environment.dotdotx, drone3.environment.dotdotx, drone4.environment.dotdotx;'. The bottom pane shows the 'Problem' view with 0 items.

```
workspace - FourDronesSystem_Formation/package/FourDronesSystem.aadl - OSATE2
File Edit Navigate Search Project Run OSATE Analyses HybridSynchAADL Window Help

AADL Navigator AADL Diagrams
FourDronesSystem_Formation
  Plug-in Contributions
  Referenced Projects
  package
    instances
      FourDronesSystem_FourDronesSystem
      Drone.aadl
      DroneControl.aadl
      Environment.aadl
      FourDronesSystem.aadl
      RefDrone.aadl
    propertysets
    requirement
      FourDronesSystem_impl_Instance.pspc
    verification

FourDronesSystem_impl_Instance.pspc
d4lv: port drone4.outVY -> drone1.inVY;

r1x: port refDrone.outX -> drone1.refX;
r2x: port refDrone.outX -> drone2.refX;
r3x: port refDrone.outX -> drone3.refX;
r4x: port refDrone.outX -> drone4.refX;
r1vx: port refDrone.outVX -> drone1.refVX;
r2vx: port refDrone.outVX -> drone2.refVX;
r3vx: port refDrone.outVX -> drone3.refVX;
r4vx: port refDrone.outVX -> drone4.refVX;

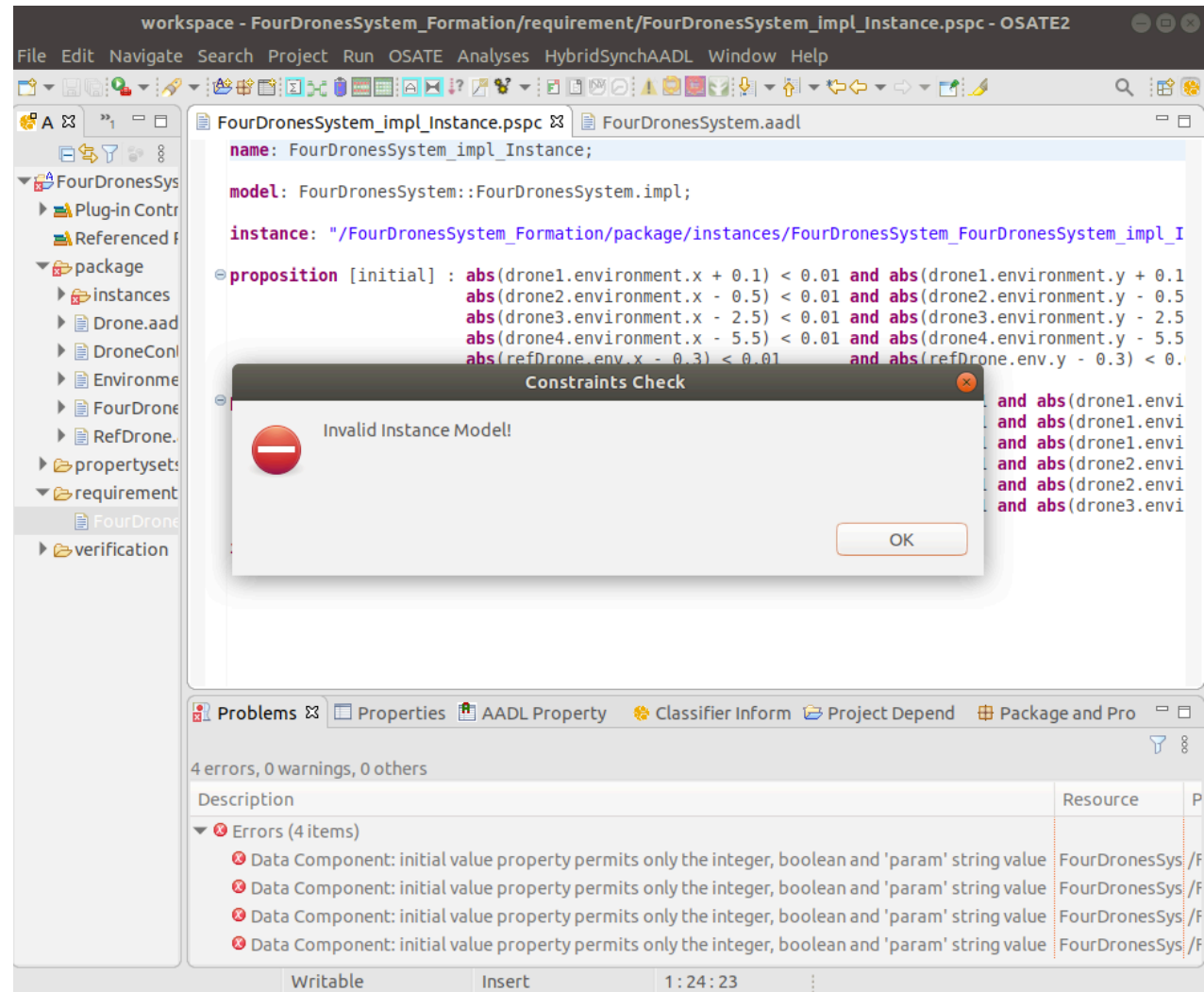
r1y: port refDrone.outY -> drone1.refY;
r2y: port refDrone.outY -> drone2.refY;
r3y: port refDrone.outY -> drone3.refY;
r4y: port refDrone.outY -> drone4.refY;
r1vy: port refDrone.outVY -> drone1.refVY;
r2vy: port refDrone.outVY -> drone2.refVY;
r3vy: port refDrone.outVY -> drone3.refVY;
r4vy: port refDrone.outVY -> drone4.refVY;

properties
Data_Model::Initial_Value => ("SomethingWrong") applies to
  drone1.environment.x, drone2.environment.x,
  drone3.environment.x, drone4.environment.x;
Data_Model::Initial_Value => ("0") applies to
  drone1.environment.dotx, drone2.environment.dotx,
  drone3.environment.dotx, drone4.environment.dotx;
Data_Model::Initial_Value => ("0") applies to
  drone1.environment.dotdotx, drone2.environment.dotdotx,
  drone3.environment.dotdotx, drone4.environment.dotdotx;

Problem Property AADL Pr Classifier Project D Package
0 items
Description Resource Path Lo
Writable Insert 57:58:1962
```


Constraints Check – Erroneous Model

- After re-instantiating the model, click **Constraints Check** to perform constraints checking.
- Click **Initial Mode**
- Our tool then shows an error message in the Problems view.

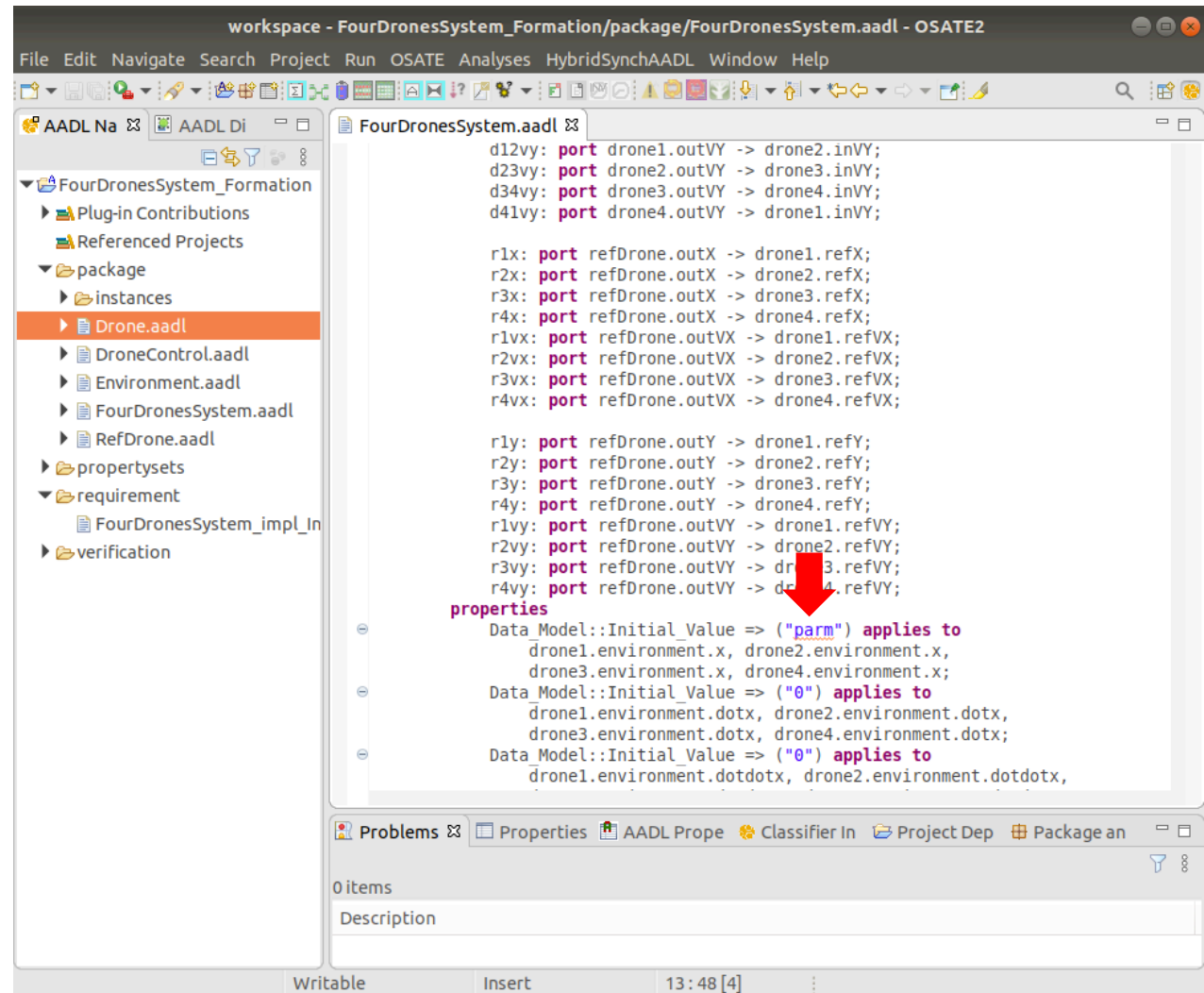


Outline

1. Basic OSATE
2. Creating Property Specification Files (PSPC)
3. HybridSynchAADL Constraints Checker
- 4. Maude Code Generation**
5. Formal Analysis

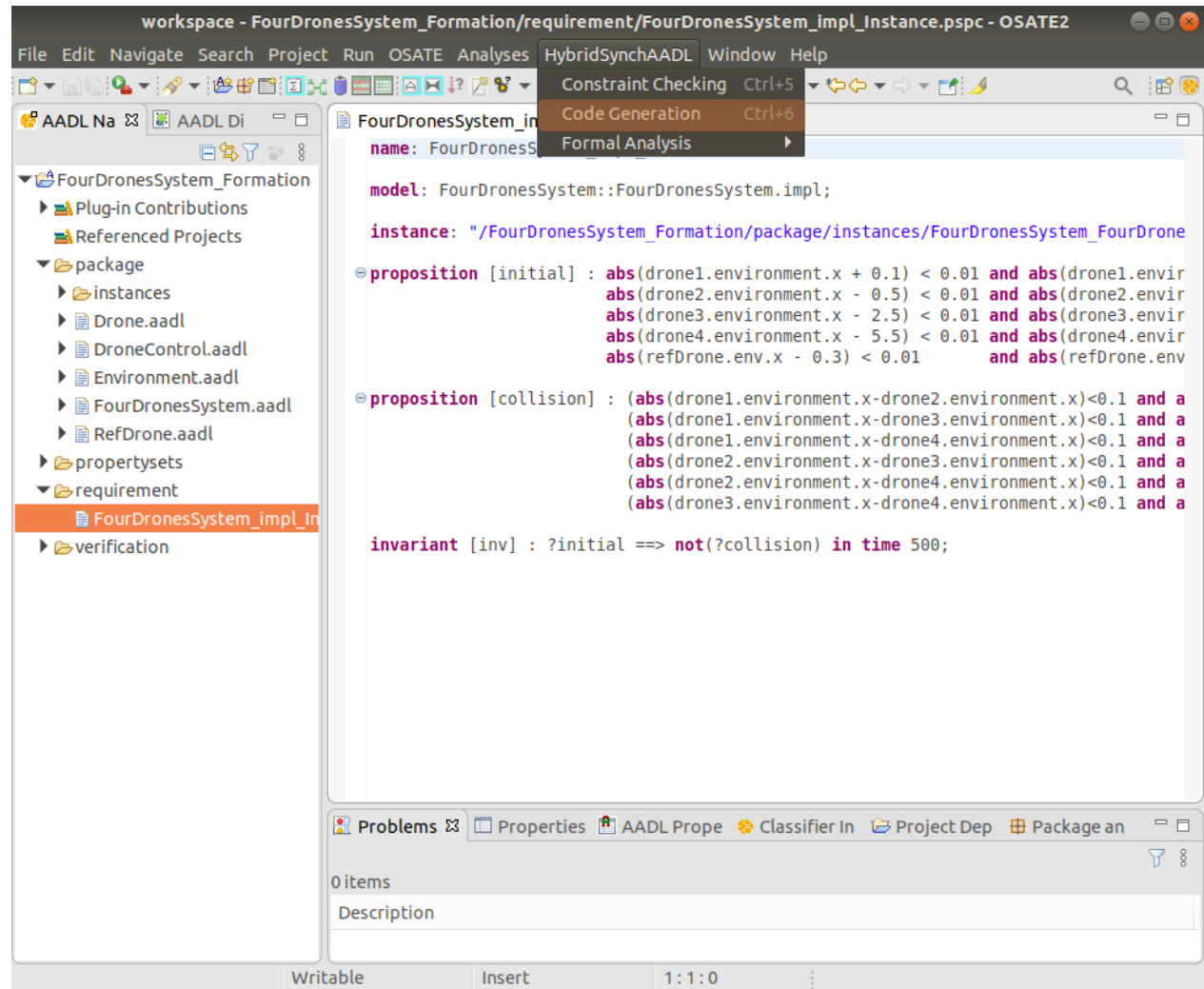
The FourDronesSystem Example

- Let us go back to the correct model.
- Don't forget to instantiate the model again.



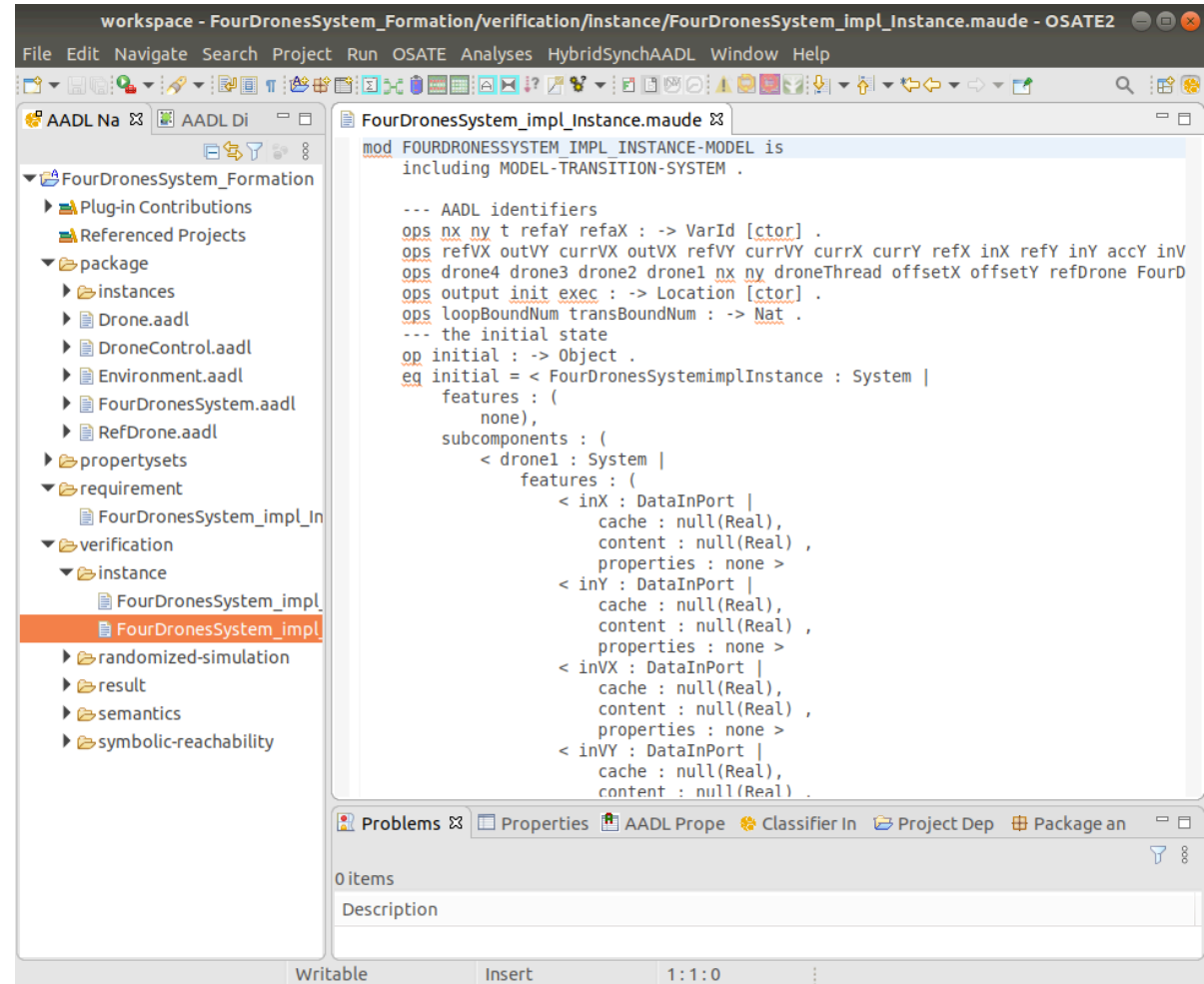
Maude Code Generation

- Click Code Generation to automatically generate the rewriting-modulo-SMT model from the HybridSynchAADL model.



Maude Code Generation

- The generated Maude files, including Maude files for properties, are in the verification/instance directory.

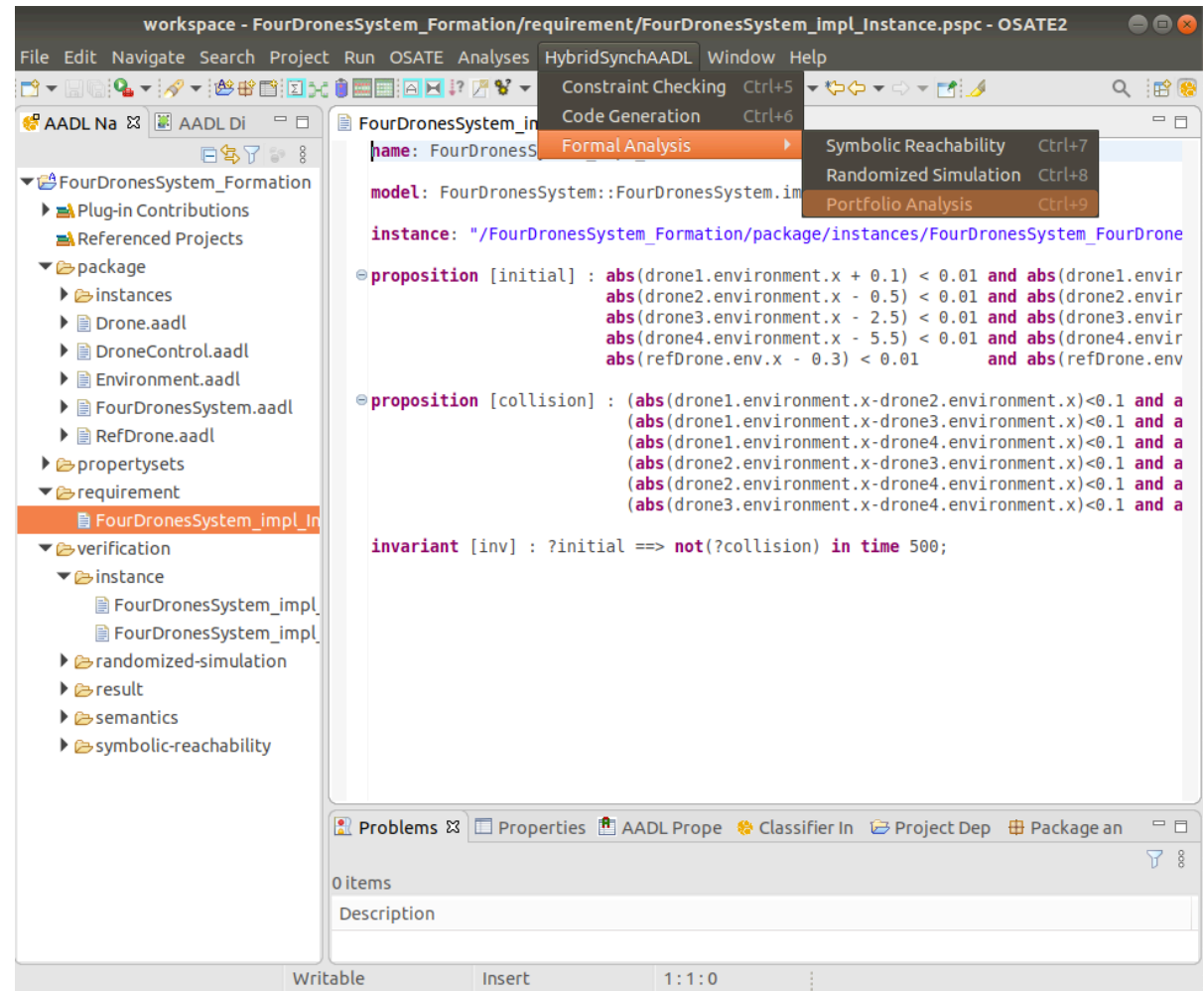


Outline

1. Basic OSATE
2. Creating Property Specification Files (PSPC)
3. HybridSynchAADL Constraints Checker
4. Maude Code Generation
- 5. Formal Analysis**

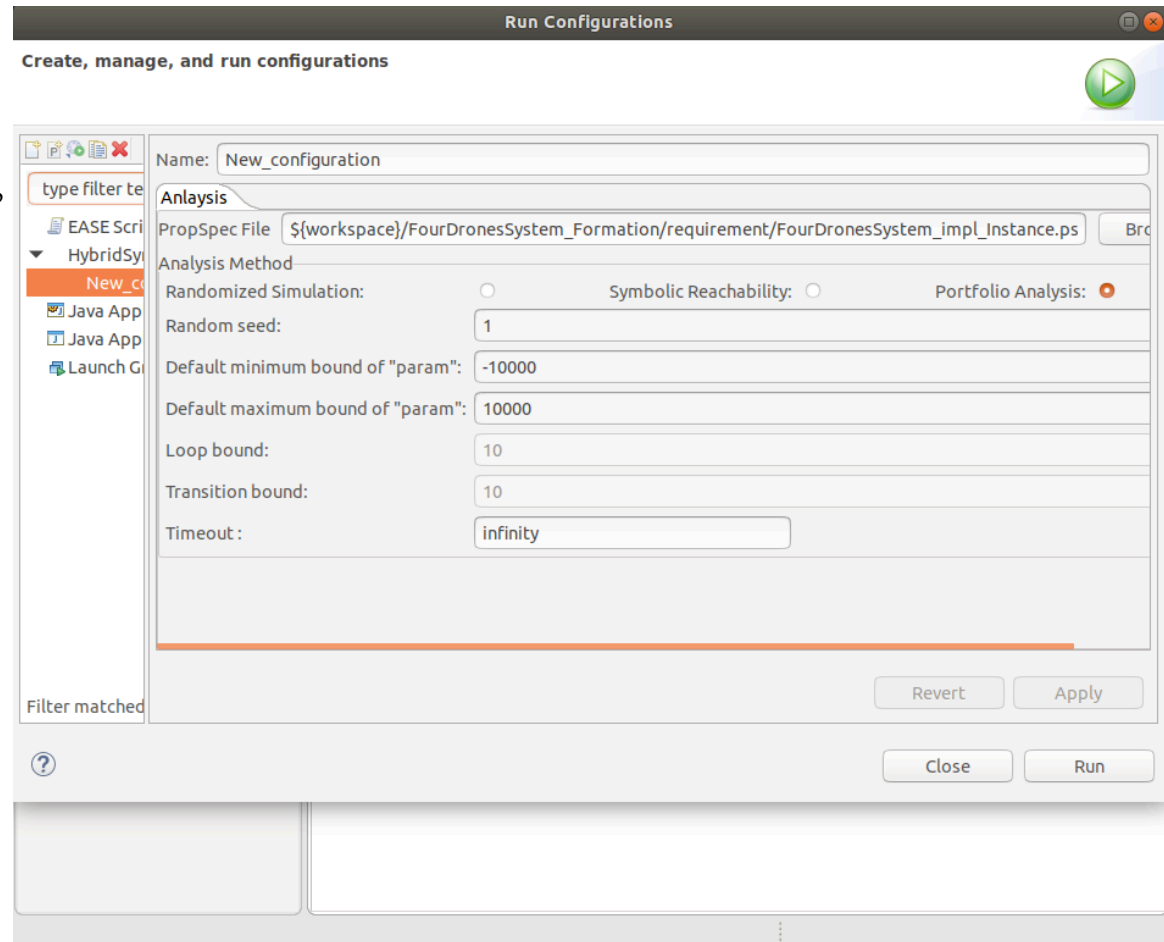
Portfolio Analysis

- Click Portfolio Analysis to perform symbolic reachability and randomized simulation simultaneously using rewriting-modulo-SMT.



Portfolio Analysis

- Create a new configuration file
- Set PSPC file
“FourDronesSystem_impl_Instance1.pspc”
path
- Click Portfolio Analysis radio button
- Set positive integer value in Timeout
 - infinity can be set for infinite time.



Analysis Results

- The HybridSynchAADL Result view shows the analysis results.

The screenshot displays the OSATE2 IDE interface. The main editor window shows the AADL code for `FourDronesSystem_impl_Instance.pspc`. The code includes a `name` declaration, a `model` declaration, an `instance` declaration, and two `proposition` declarations (one for initial state and one for collision), along with an `invariant` declaration. The bottom panel shows the `HybridSynch` results table, which contains the following data:

PSPC File	Property Id	Result	Method	CPUTime	RunningTime	Location
FourDronesSystem_impl_Instance.pspc	inv	Counterexample found	random	196ms	2165ms	/Fo

A red arrow points to the `HybridSynch` tab in the bottom panel.

Counterexample

- Each file in Location in the result view contains a counterexample of an invariant property if it exists.

The screenshot shows the workspace with the file `FourDronesSystem_impl_Instance.pspc` open. The code defines a system with four drones, each with a `droneProc` and `droneThread`. The `droneThread` for each drone is defined with variables `offsetX`, `offsetY`, `refVX0`, and `refVY0`. The `droneThread` for drone4 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone3 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone2 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone1 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone0 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone4 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone3 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone2 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone1 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone0 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`.

PropertyId	Result	Method	CPUTime	RunningTime	Location
l_Instance.pspc inv	Counterexample found	random	196ms	2165ms	/FourDronesSystem_Forma

The screenshot shows the workspace with the file `FourDronesSystem_impl_Instance.pspc` open. The code defines a system with four drones, each with a `droneProc` and `droneThread`. The `droneThread` for each drone is defined with variables `offsetX`, `offsetY`, `refVX0`, and `refVY0`. The `droneThread` for drone4 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone3 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone2 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone1 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`. The `droneThread` for drone0 is defined with variables `offsetX`, `offsetY`, `refVY0`, and `refVX0`.

PropertyId	Result	Method	CPUTime	RunningTime	Location
l_Instance.pspc inv	Counterexample found	random	196ms	2165ms	/FourDronesSystem_Forma

Thank you!