

HybridSynchAADL

Tutorial

Outline

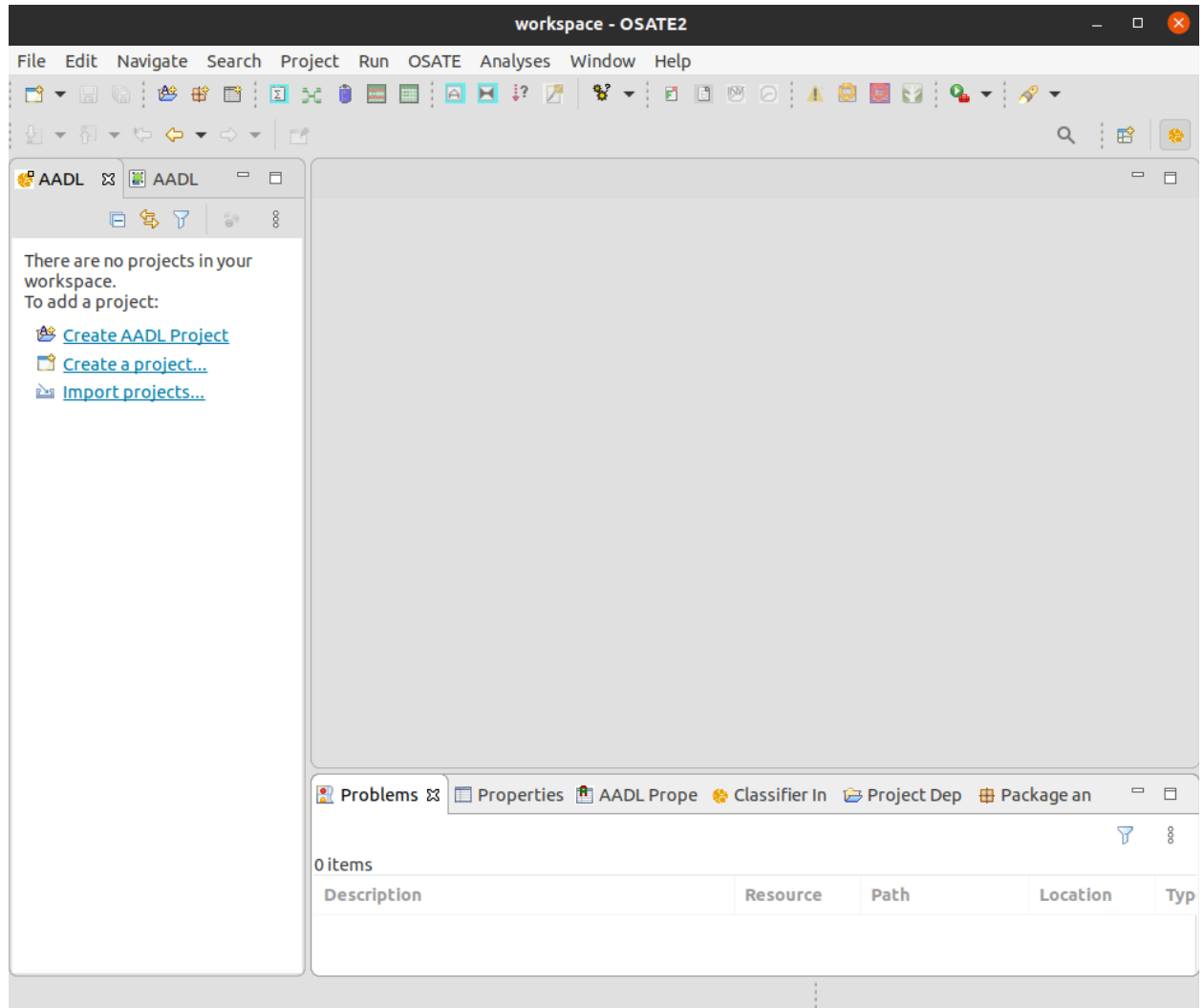
1. Basic OSATE
2. Creating Property Specification Files (PSPC)
3. HybridSynchAADL Constraints Checker
4. Maude Code Generation
5. Formal Analysis

Outline

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

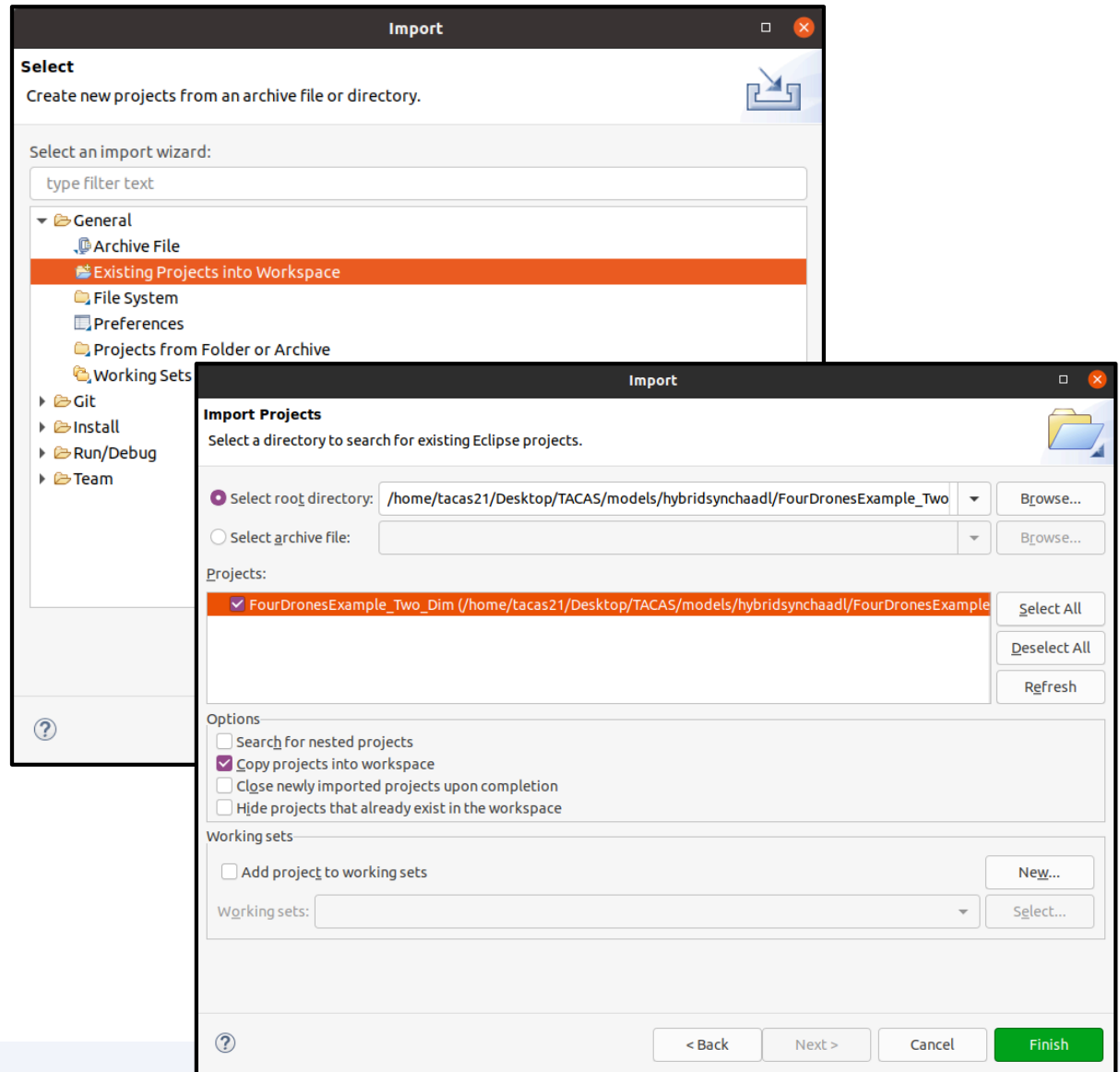
Running OSATE

- You will see this window when you execute OSATE.



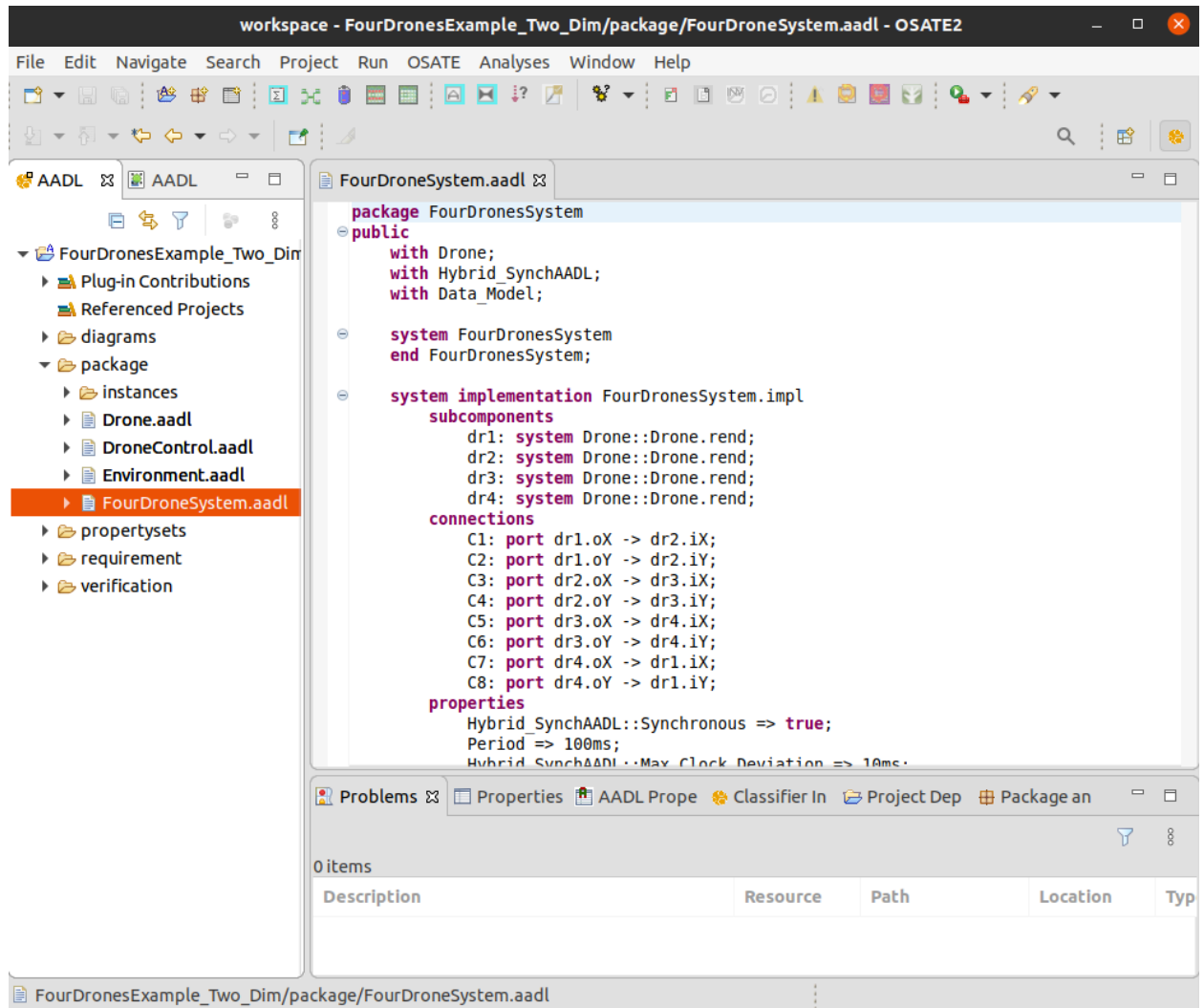
OSATE - Importing an Example

- We start with a simple example, namely, `FourDronesSystem2` in the directory `models/hybridsynchaad1`.
- To import the example, choose
 - Menu → File → Import → General → Existing Projects into Workspace.



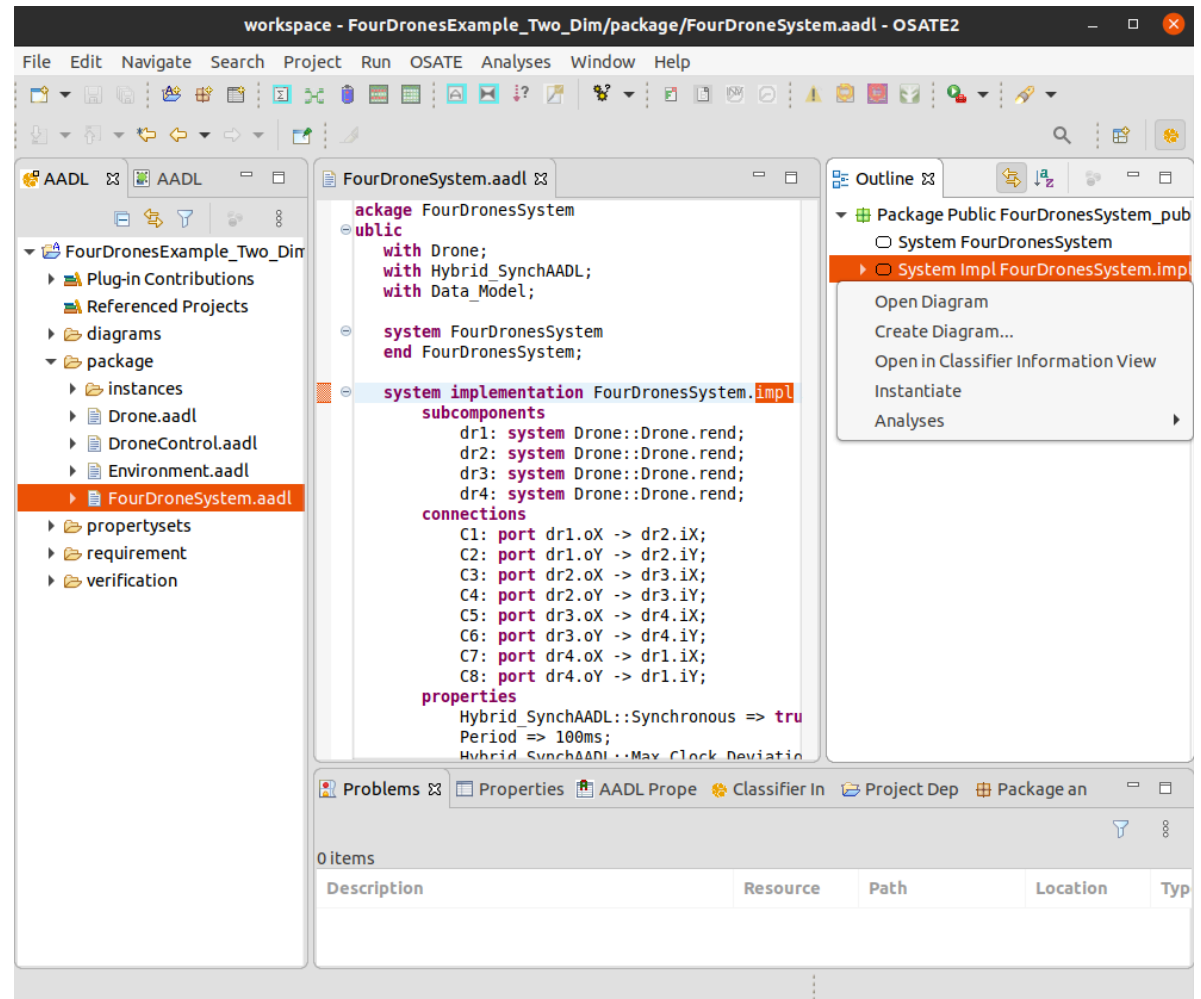
FourDronesSystem – Text

- `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.

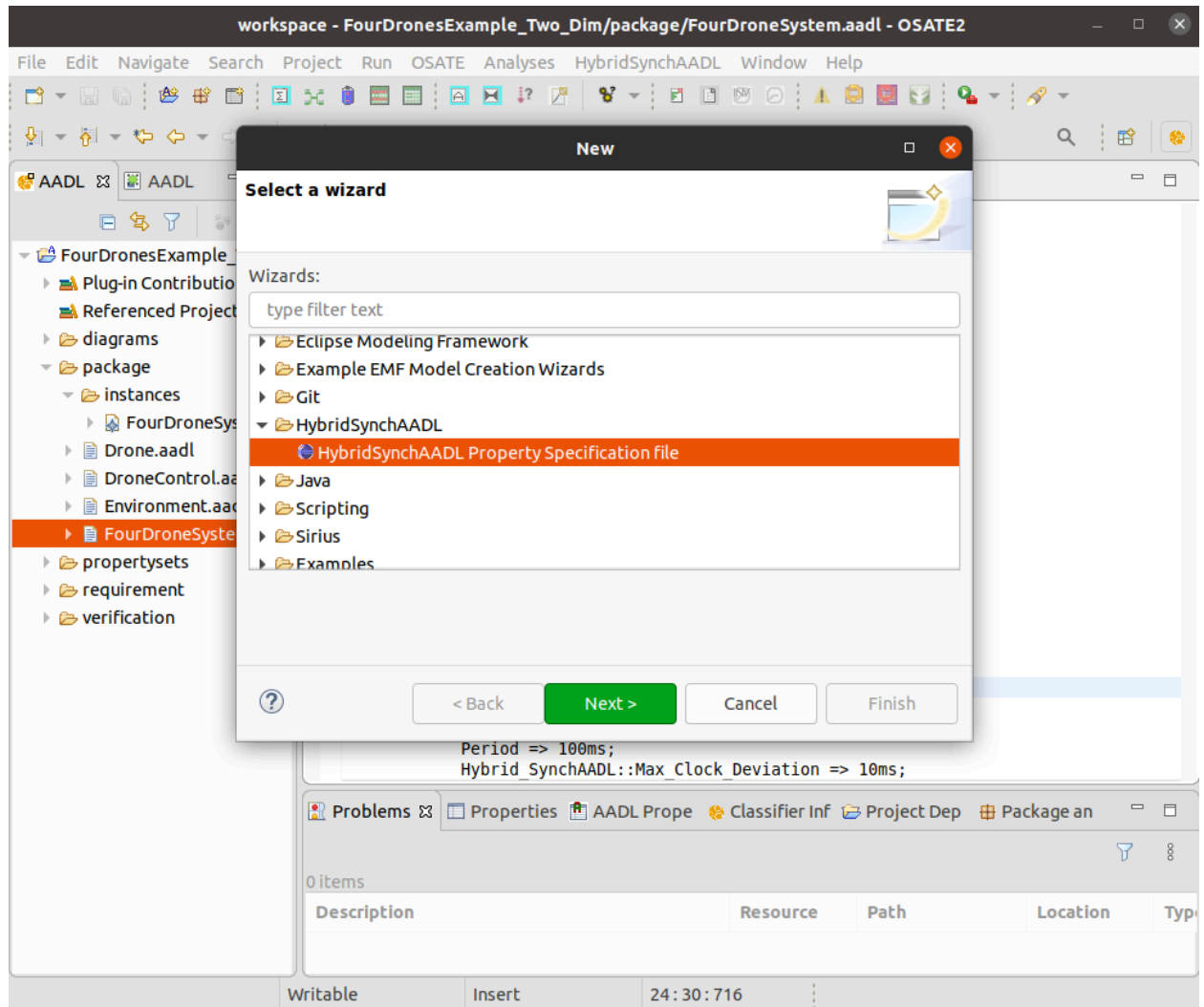


Outline

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

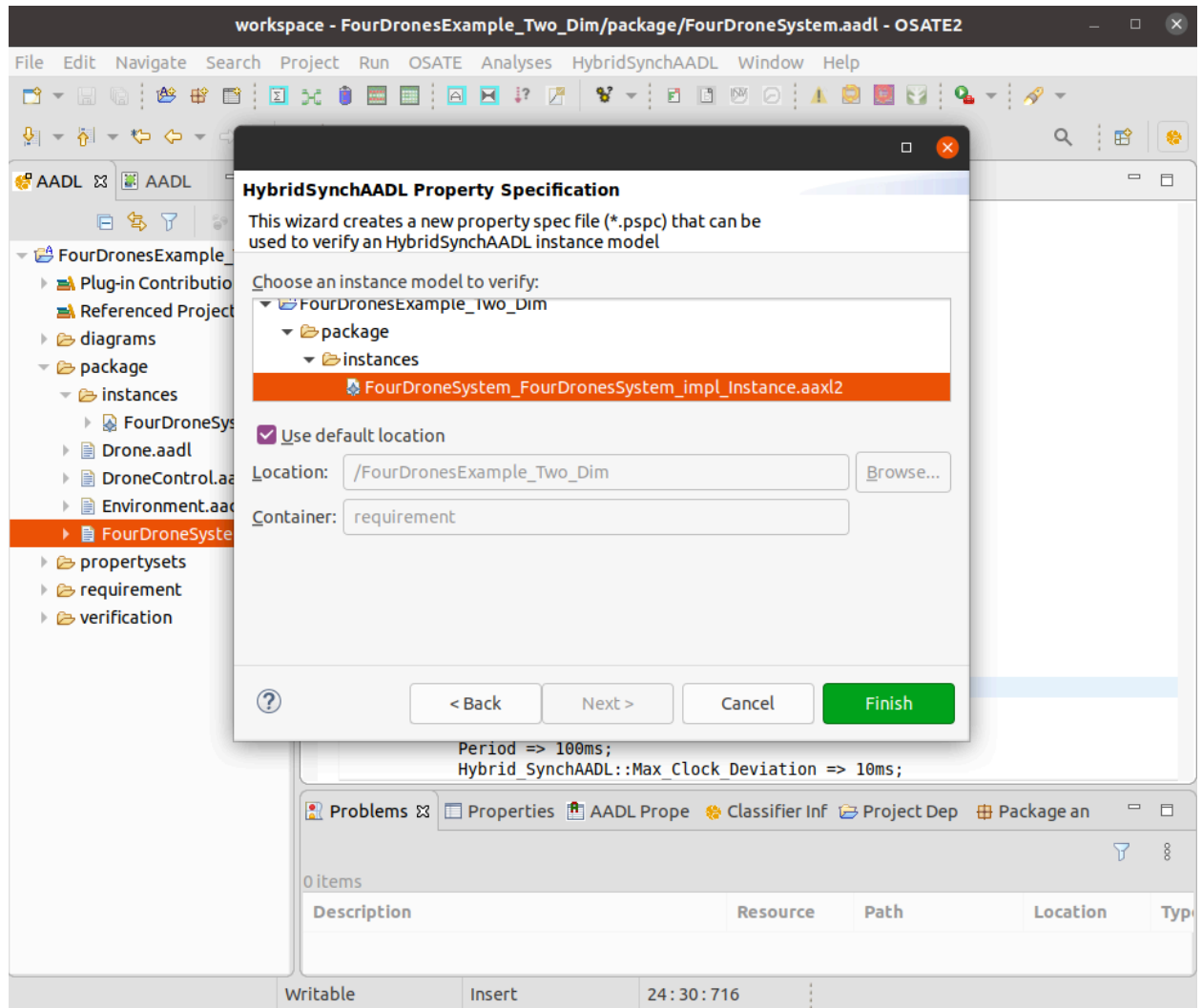
Creating PSPC Files

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



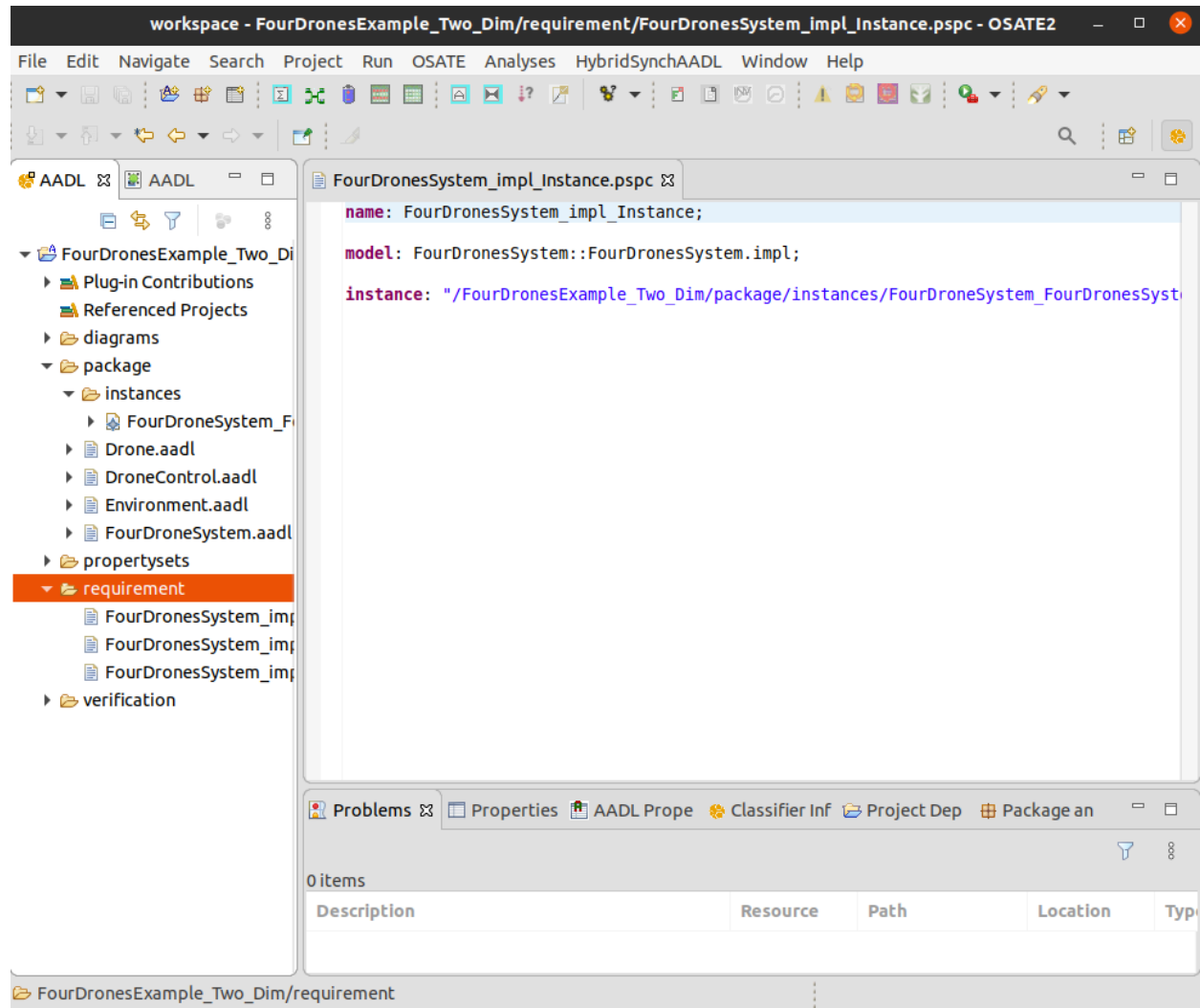
Creating PSPC Files

- 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 two sample PSPC files in this project.

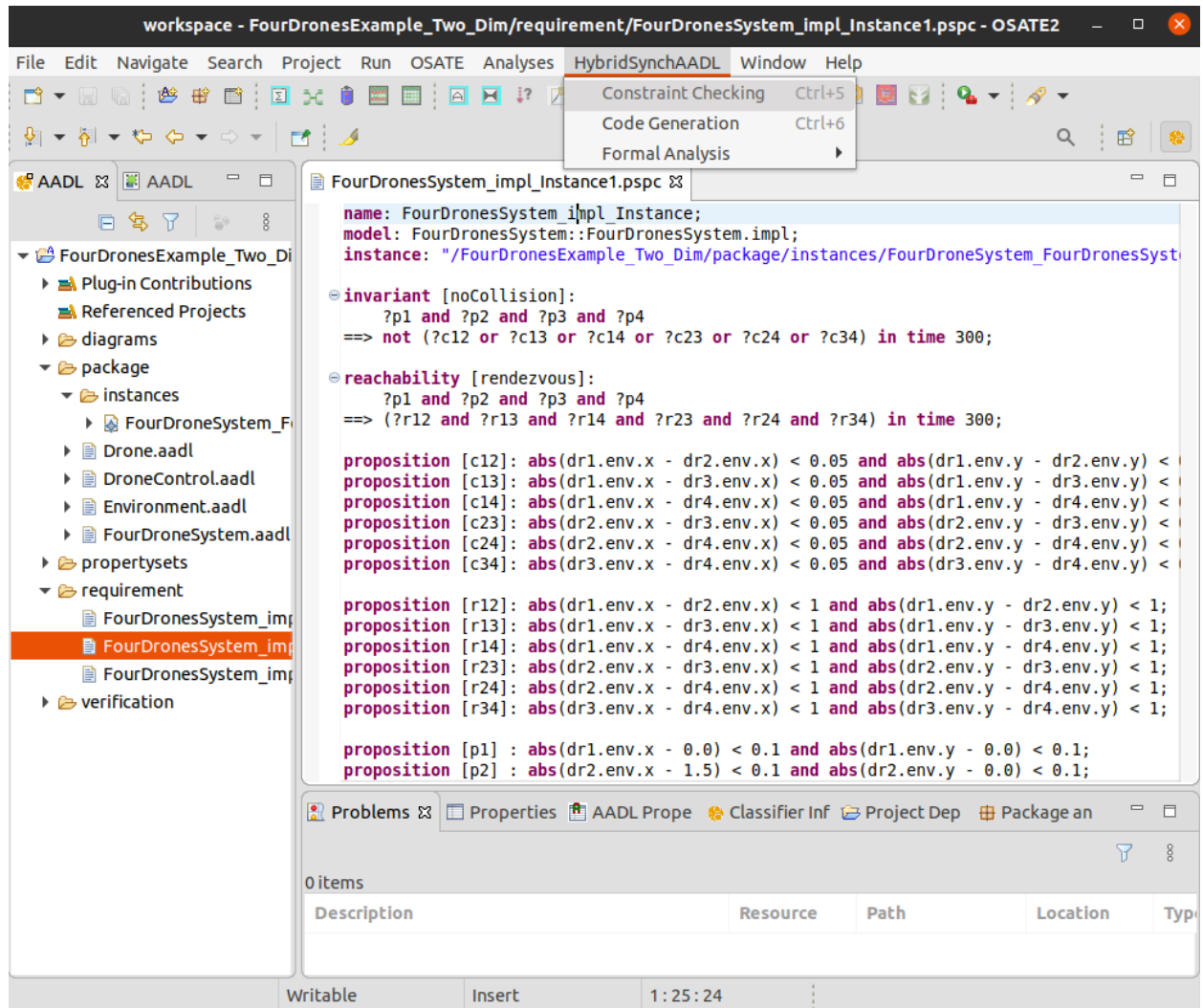


Outline

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

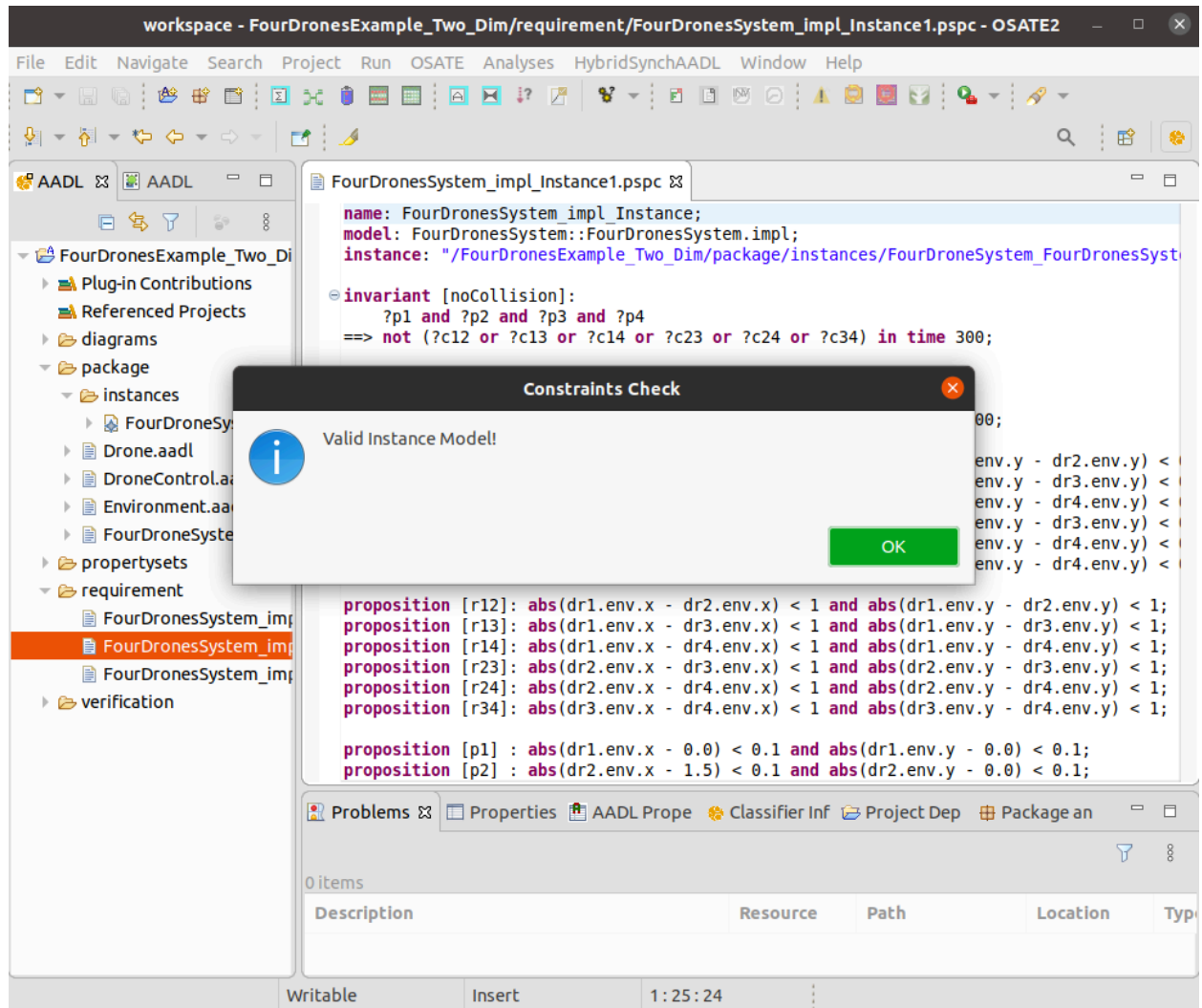
Checking HybridSynchAADL Constraints

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



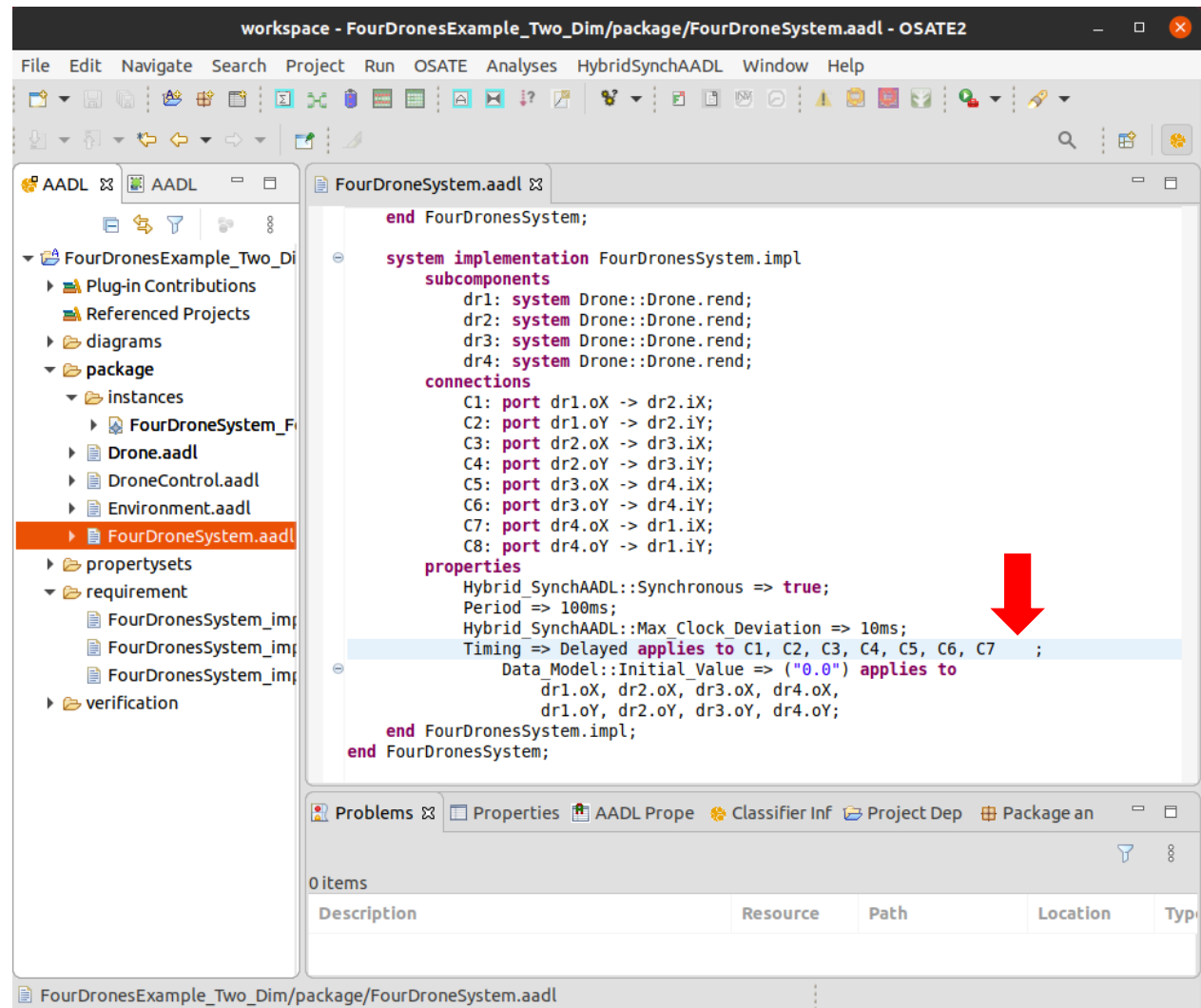
Checking HybridSynchAADL Constraints

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



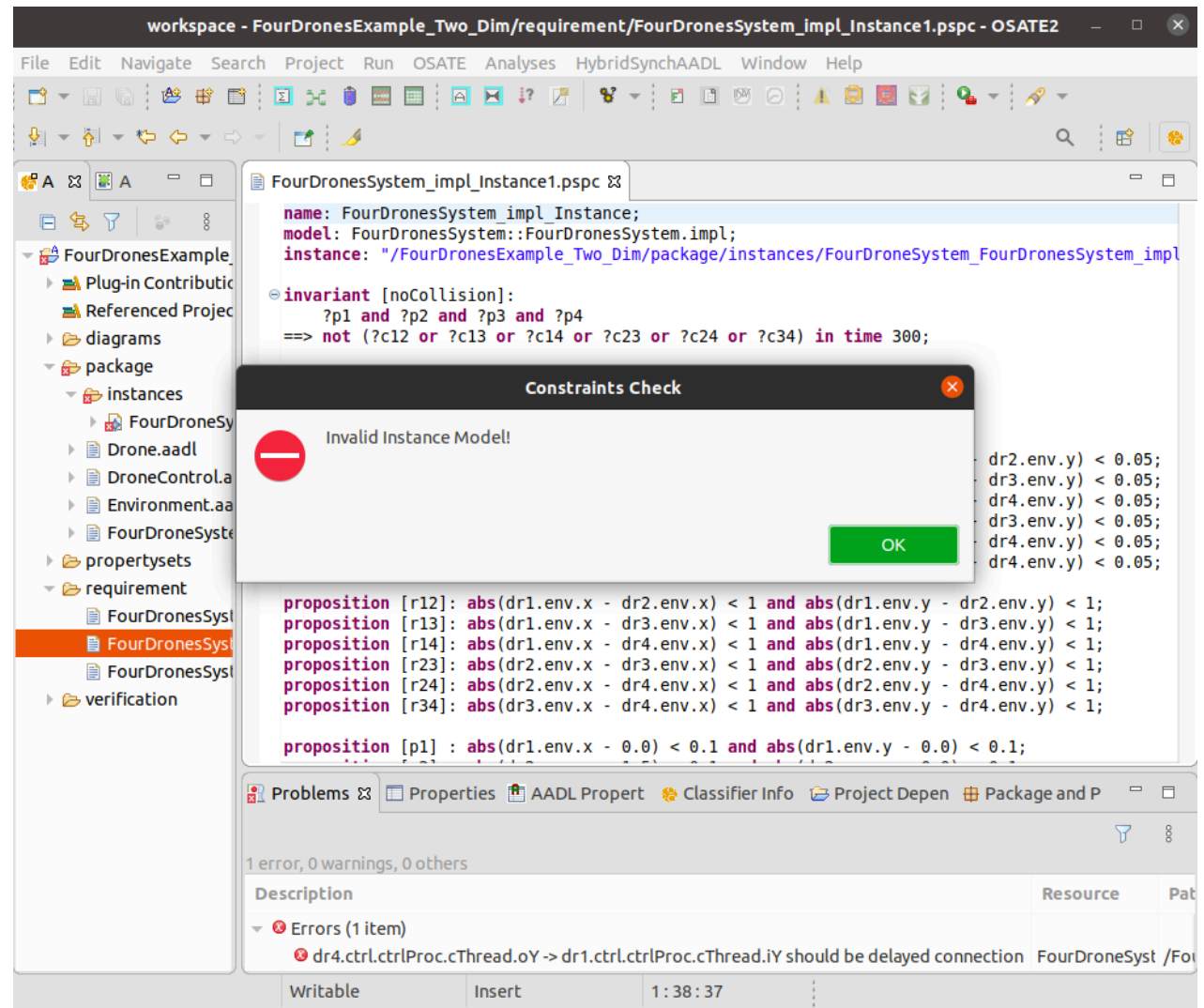
Constraints Check – Erroneous Model

- What if some HybridSynchAADL constraints is not satisfied?
- Let us add an invalid immediate connection and see what happened.
 - by removing the property Timing => Delayed from the connection C8.



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.

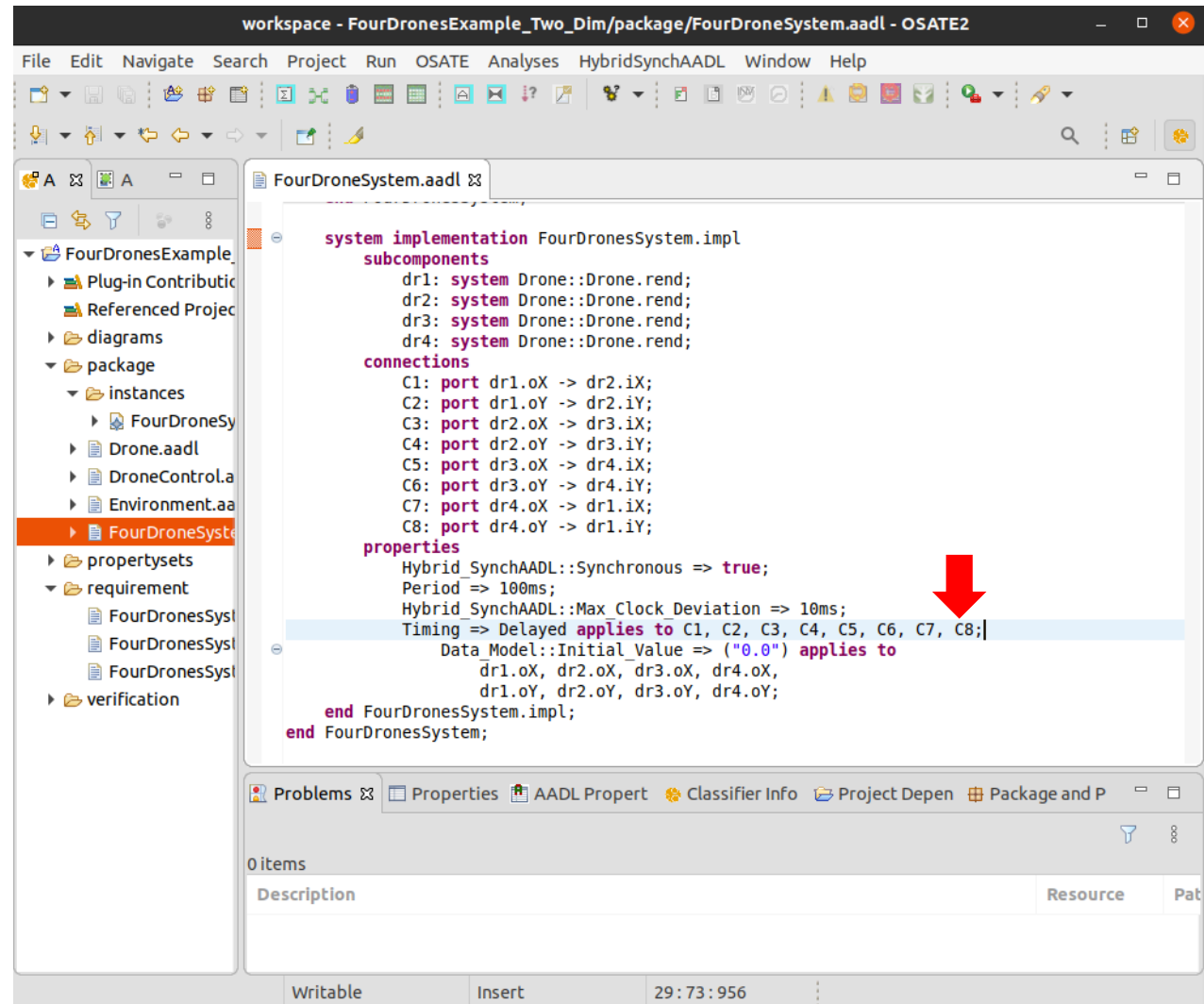


Outline

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

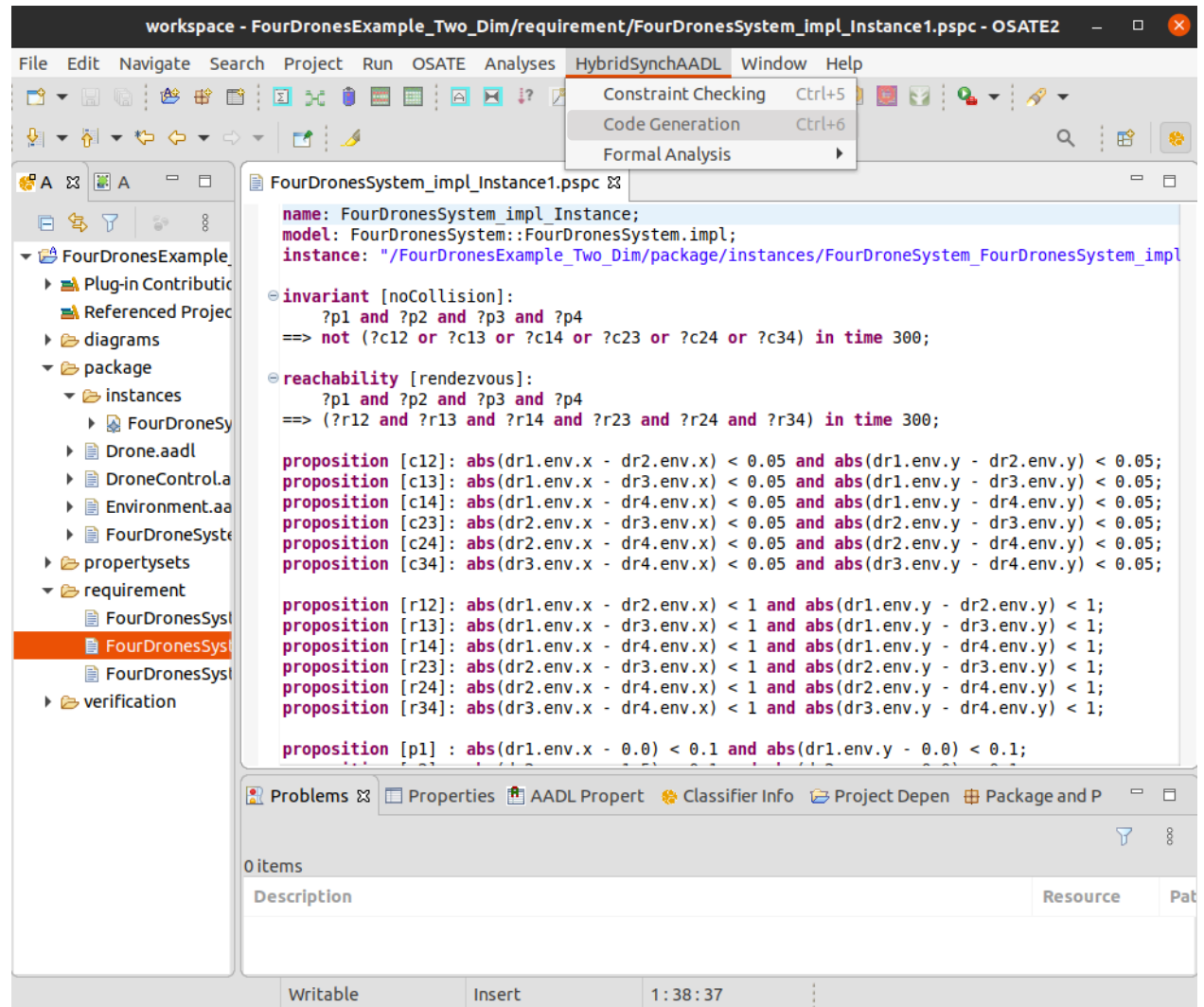
The FourDronesSystem Example

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



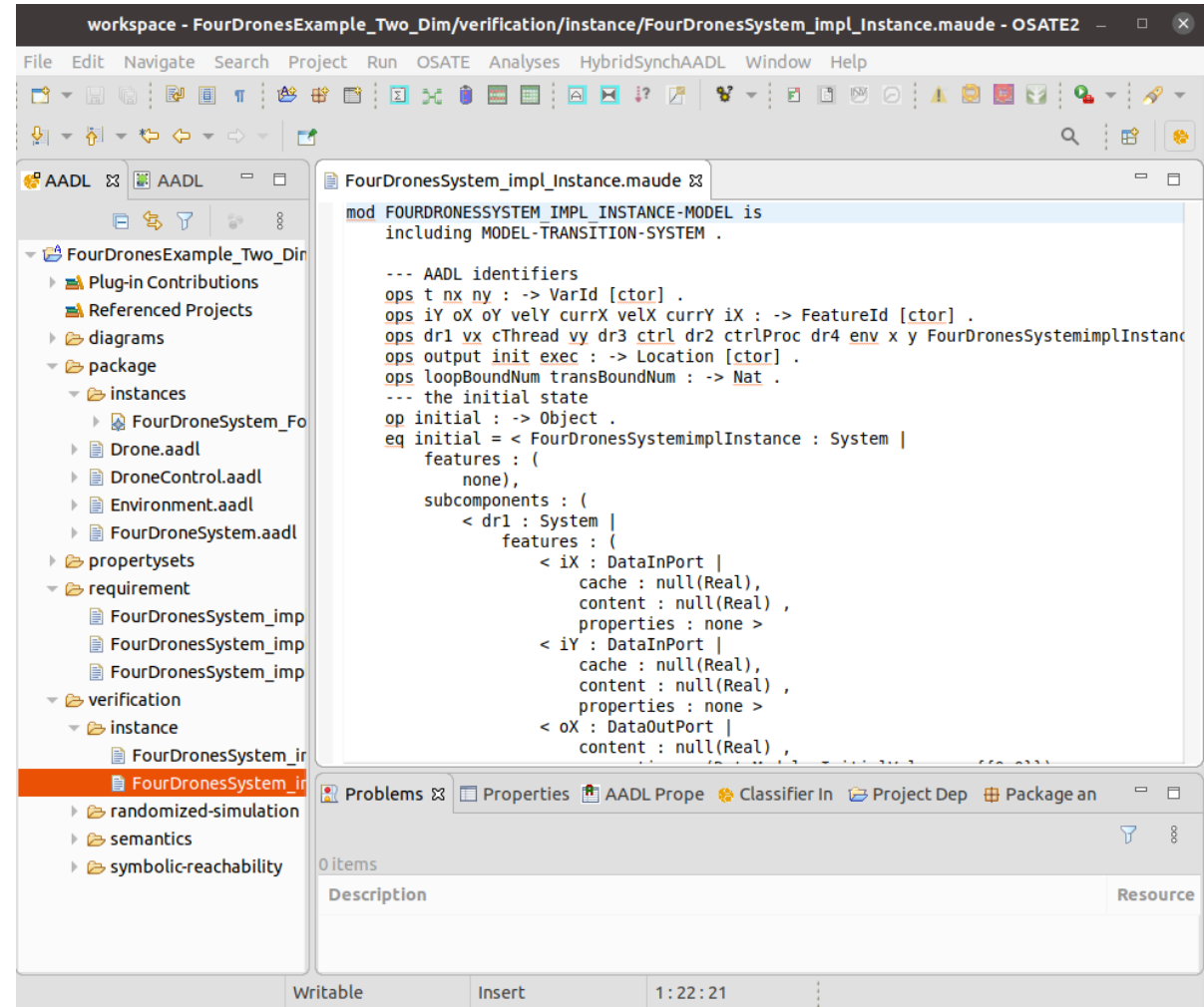
Maude Code Generation

- 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.

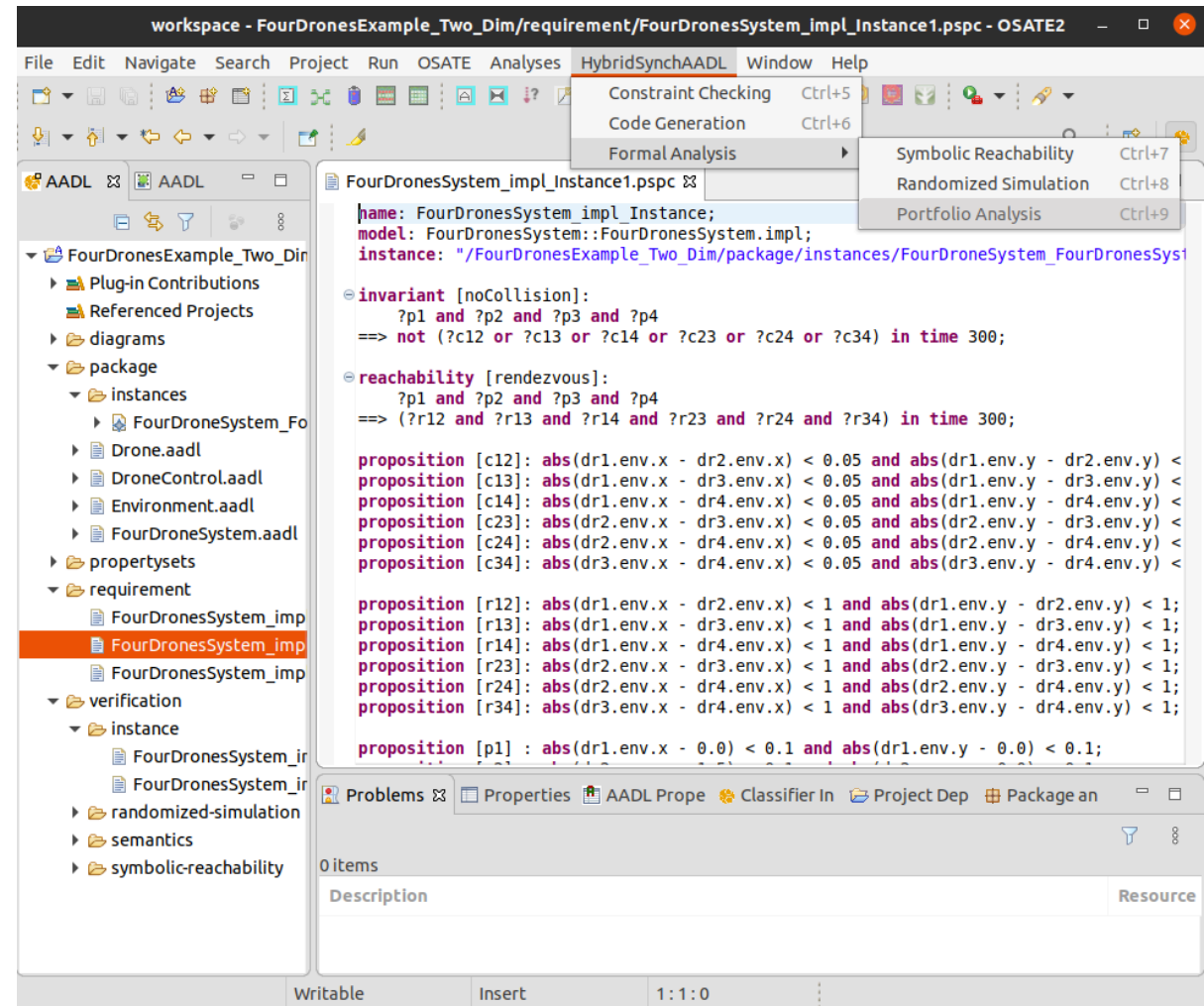


Outline

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

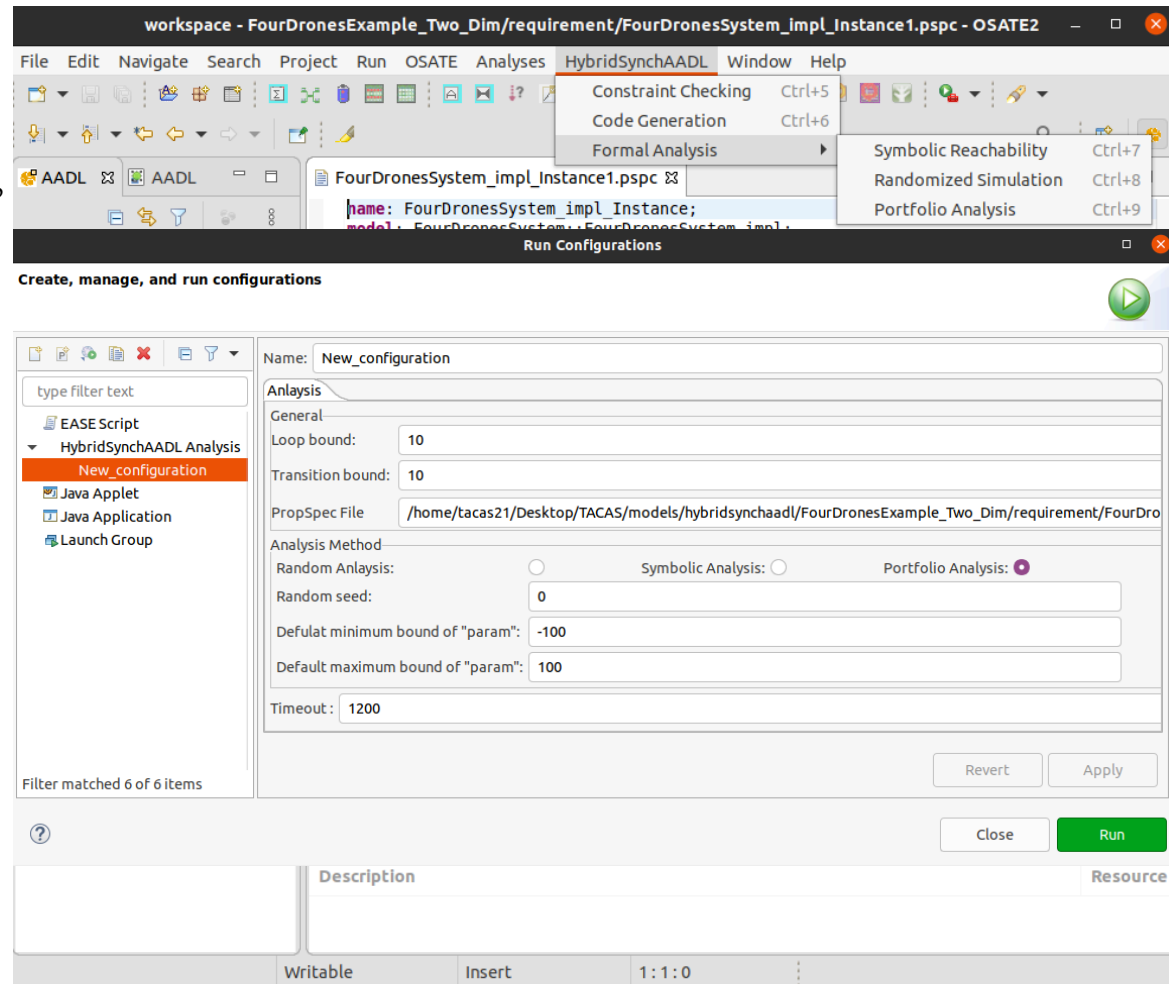
Portfolio Analysis

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



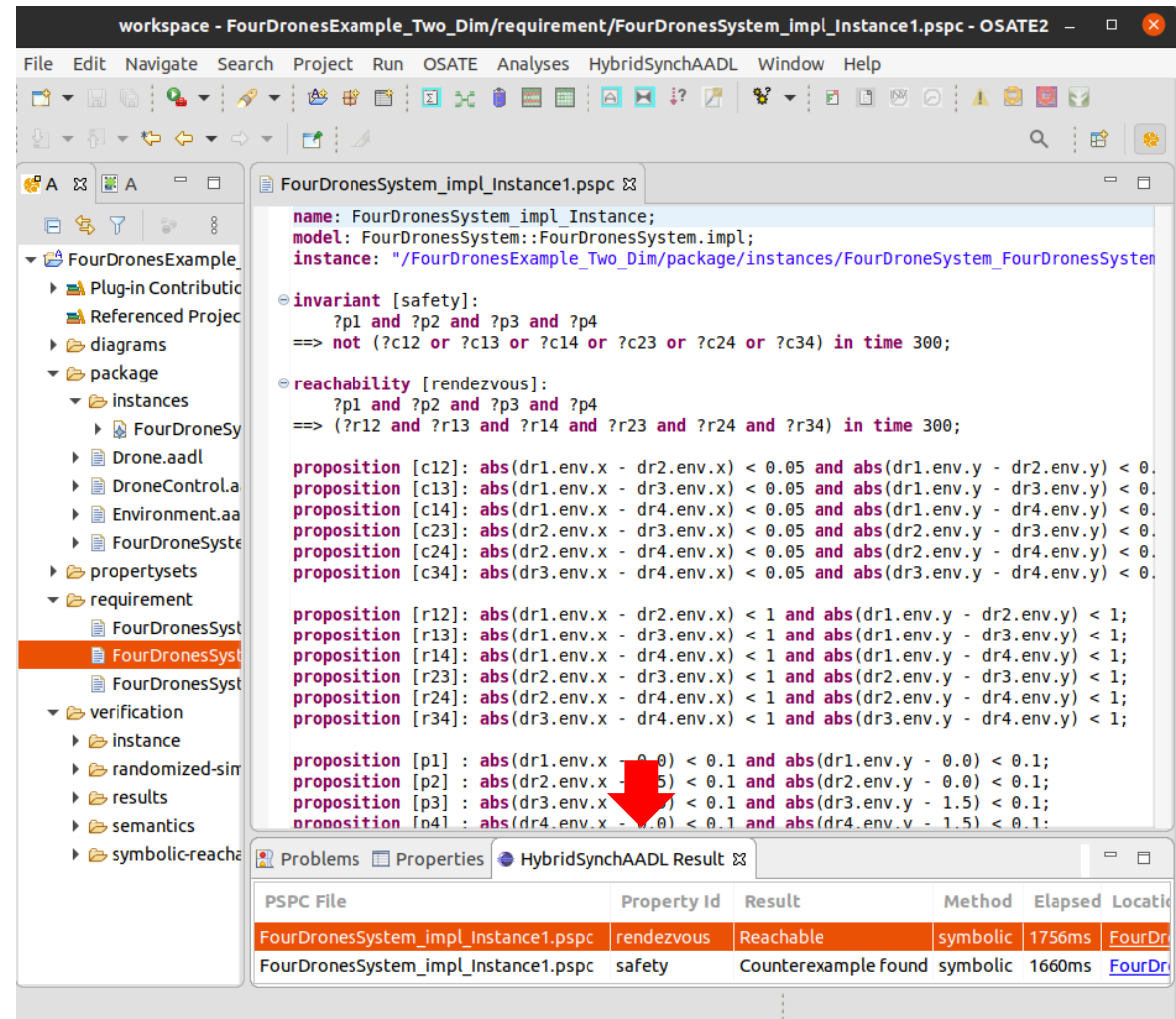
Portfolio Analysis

- Create a new configuration file
- Set PSPC file
“FourDronesSystem_impl_Instance1.pspc”
path
- Click Portfolio Analysis radio button
- Set positive integer value in Random Seed
- Set proper range value for parameterized variables.
- Set positive integer value in Timeout
 - -1 can be set for infinite time.



Analysis Results (1)

- The HybridSynchAADL Result view shows the analysis results.



The screenshot shows the HybridSynchAADL Result view in a workspace. The main editor displays the source code for `FourDronesSystem_impl_Instance1.pspc`, which includes several propositions and invariants. A red arrow points to the proposition `proposition [p1] : abs(dr1.env.x - 0.0) < 0.1 and abs(dr1.env.y - 0.0) < 0.1;`. The bottom panel shows the 'HybridSynchAADL Result' table, which contains the following data:

PSPC File	Property Id	Result	Method	Elapsed	Location
FourDronesSystem_impl_Instance1.pspc	rendezvous	Reachable	symbolic	1756ms	FourDr
FourDronesSystem_impl_Instance1.pspc	safety	Counterexample found	symbolic	1660ms	FourDr

Counterexamples and Witnesses

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

The screenshot displays the HybridSynchronAADL IDE interface. The main editor window shows the file `FourDronesSystem_impl_Instance1-symbolic-rendezvous.txt` with the following content:

```
Time: 0
FourDronesSystemrendInstance ->[
  dr1 ->[
    (ctrl . ctrlProc . cThread) ->[
      variables: none
      currState: 'init'
    env ->[
      variables: (vx |>= -9.0),(vy |>= 2.0e+1),(x |>= 0.0),y |>= 0.0
      currMode: '@@default@loc@@]
  dr3 ->[
    (ctrl . ctrlProc . cThread) ->[
      variables: none
      currState: 'init'
    env ->[
      variables: (vx |>= -2.8e+1),(vy |>= -1.05e+2),(x |>= 1.5),y |>= 1.5
      currMode: '@@default@loc@@]
  dr2 ->[
    (ctrl . ctrlProc . cThread) ->[
      variables: none
      currState: 'init'
    env ->[
      variables: (vx |>= -9.8e+1),(vy |>= 3.1e+1),(x |>= 1.5),y |>= 0.0
      currMode: '@@default@loc@@]
  dr4 ->[
    (ctrl . ctrlProc . cThread) ->[
      variables: none
      currState: 'init'
    env ->[
      variables: (vx |>= 2.2e+1).(vy |>= -7.4e+1).(x |>= 0.0).v |>= 1.5
```

The left sidebar shows a project tree with the following structure:

- FourDronesExample
 - Plug-in Contributic
 - Referenced Project
 - diagrams
 - package
 - instances
 - FourDroneSy
 - Drone.aadl
 - DroneControl.a
 - Environment.aa
 - FourDroneSyste
 - propertysets
 - requirement
 - FourDronesSyst
 - FourDronesSyst
 - FourDronesSyst
 - verification
 - instance
 - randomized-sim
 - results
 - FourDronesS
 - FourDronesS
 - FourDronesS
 - FourDronesS
 - semantics
 - symbolic-reacha

The bottom panel shows the 'HybridSynchronAADL Result' view with the following table:

PSPC File	Property Id	Result	Method	Elapsed	Location
FourDronesSystem_impl_Instance1.pspc	rendezvous	Reachable	symbolic	1756ms	FourDr
FourDronesSystem_impl_Instance1.pspc	safety	Counterexample found	symbolic	1660ms	FourDr

The bottom status bar shows the following information:

Writable Insert 1:1:0

Analysis Results (2)

- In the case of “FourDronesSystem_impl_Instance2.pspc”
- The result shows there is no counterexample found and a reachability of witness found

The screenshot shows the OSATE2 workspace for the project "FourDronesExample_Two_Dim/requirement/FourDronesSystem_impl_Instance2.pspc - OSATE2". The left sidebar shows the project structure, including the "FourDronesSystem_impl_Instance2.pspc" file. The main editor displays the source code for "FourDronesSystem_impl_Instance2.pspc", which includes several propositions and invariants. The bottom panel shows the "HybridSynchAADL Result" table, which contains the analysis results for the file.

PSPC File	Property Id	Result	Method	Elapsed	Location
FourDronesSystem_impl_Instance2.pspc	rendezvous	Reachable	symbolic	18004ms	FourDronesSystem_impl_Instance2.pspc
FourDronesSystem_impl_Instance2.pspc	safety	No counterexample found	symbolic	16928ms	FourDronesSystem_impl_Instance2.pspc

Thank you!