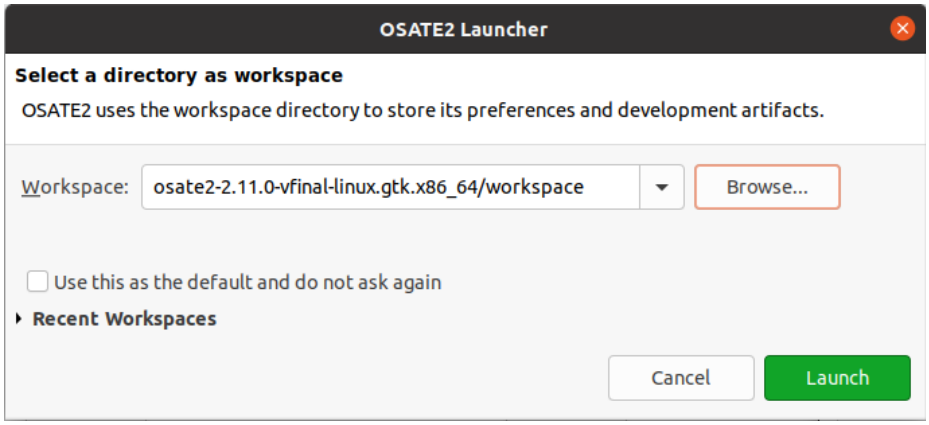
Artifact Availability

Eclipse products for the Windows and Linux platforms into which OSATE and OSATE-DIM have been installed are provided in a Zenodo repository (DOI: 10.5281/zenodo.6971721). In the provided archive file, the *eclipse* directory contains the Eclipse product and the *workspace* directory contains our test project as an artifact supporting the claims in the paper.

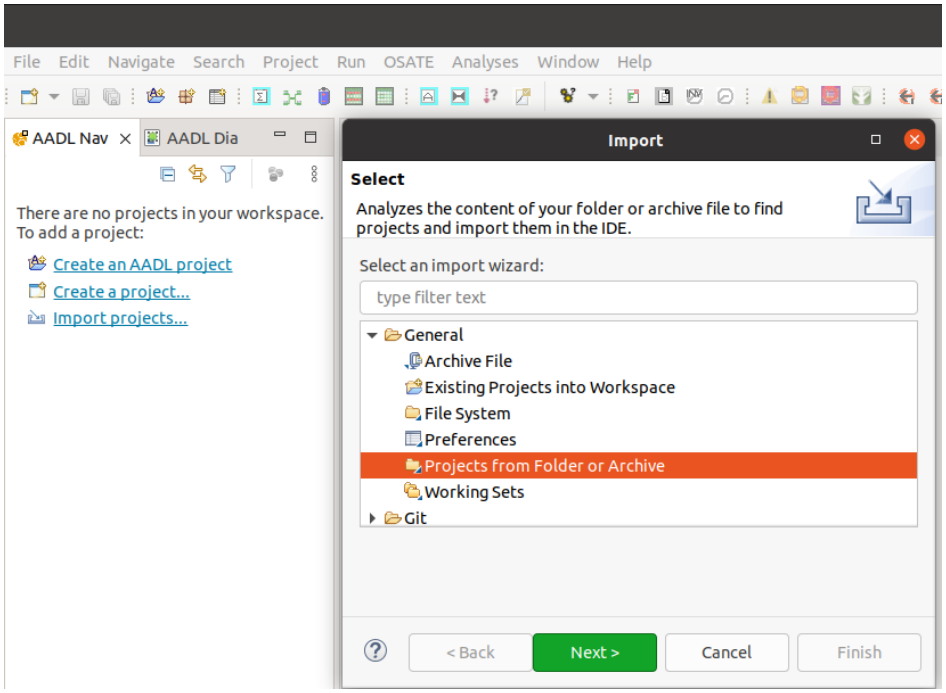
The steps to set-up the OSATE-DIM testing environment are as follows:

1. Unzip the archive.
2. Initialize the Eclipse IDE by clicking on the 'osate' executable file within the osate directory of the archive.
3. You will be prompted by the OSATE2 Launcher, to select a workspace: Please select the *workspace* directory which is provided within the archive.

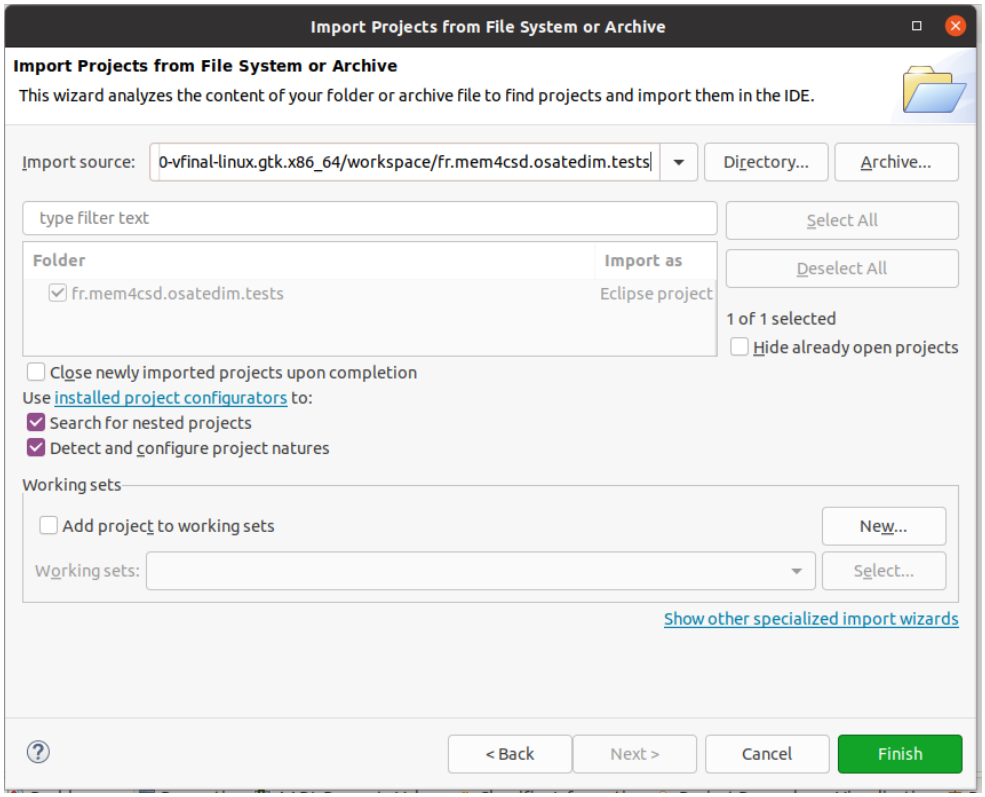


4. In the Eclipse IDE, within the AADL Perspective, you will import the test project by clicking 'Import Projects…' in the AADL Navigator or by clicking menu 'File>>Import…'.

In the Import wizard, select ‘General >> Projects from Folder or Archive’. Click ‘Next‘.



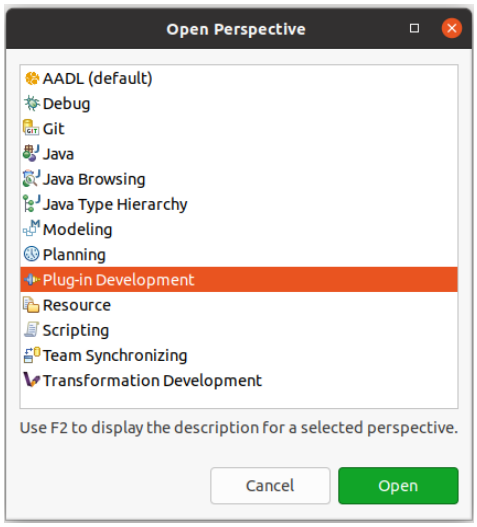
5. In the ‘Import Projects from File System or Archive’ dialog, specify the Import source as a Directory. The directory is contained within the workspace directory. It will look like this:



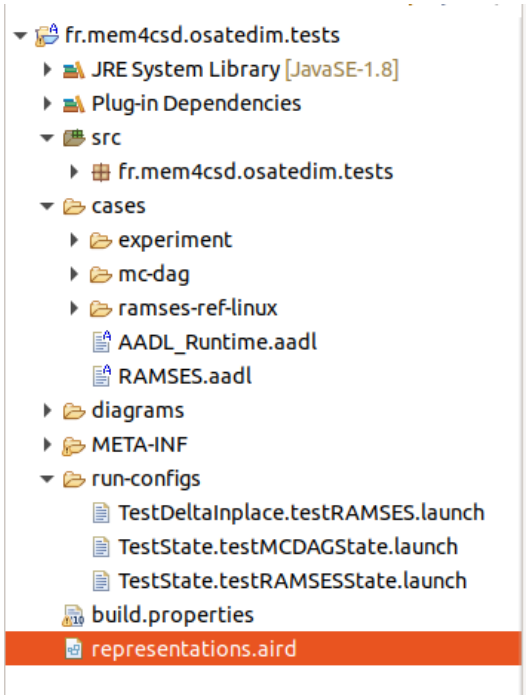
6. Click ‘Finish’.
7. You will now see the ‘fr.mem4csd.osatedim.tests’ project within the AADL Navigator.

Reproducing Results

1. Open the Plug-in Development Perspective by going to ‘Window >> Perspective >> Open Perspective’. Select ‘Plug-in Development’ in the ‘Open Perspective’ dialog.

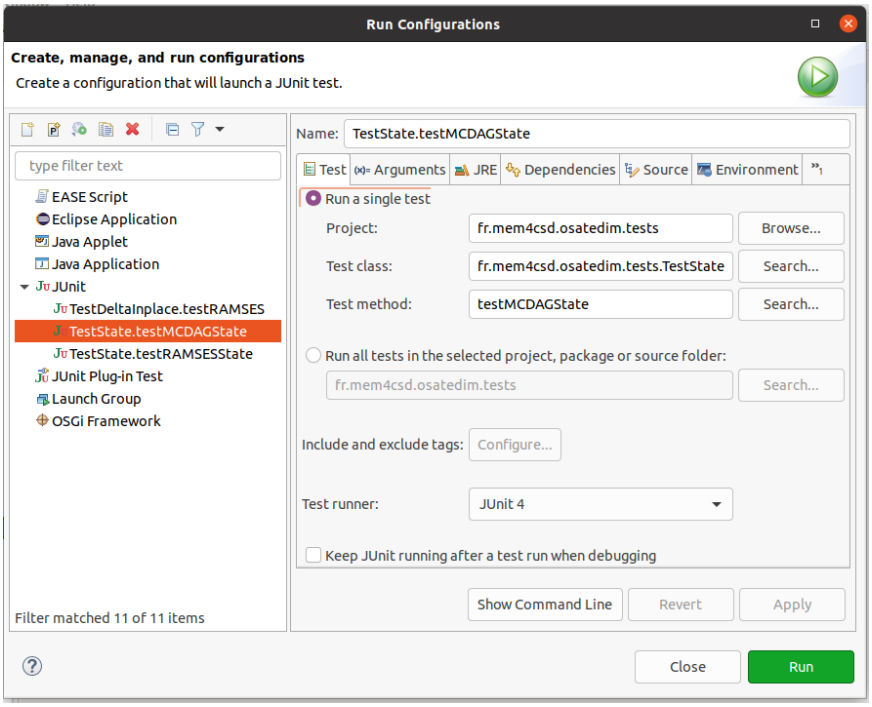


2. Unfold the ‘fr.mem4csd.osatedim.tests’ project and observe its structure:



   a. The src/ package contains the source code responsible for interfacing the examples with the OSATE-DIM plug-in. It contains the classes and methods which perform the changes/refinements on the instance models. Users can observe each particular change and its program within this package, to also see how OSATE-DIM can be interfaced with other AADL-based tools.

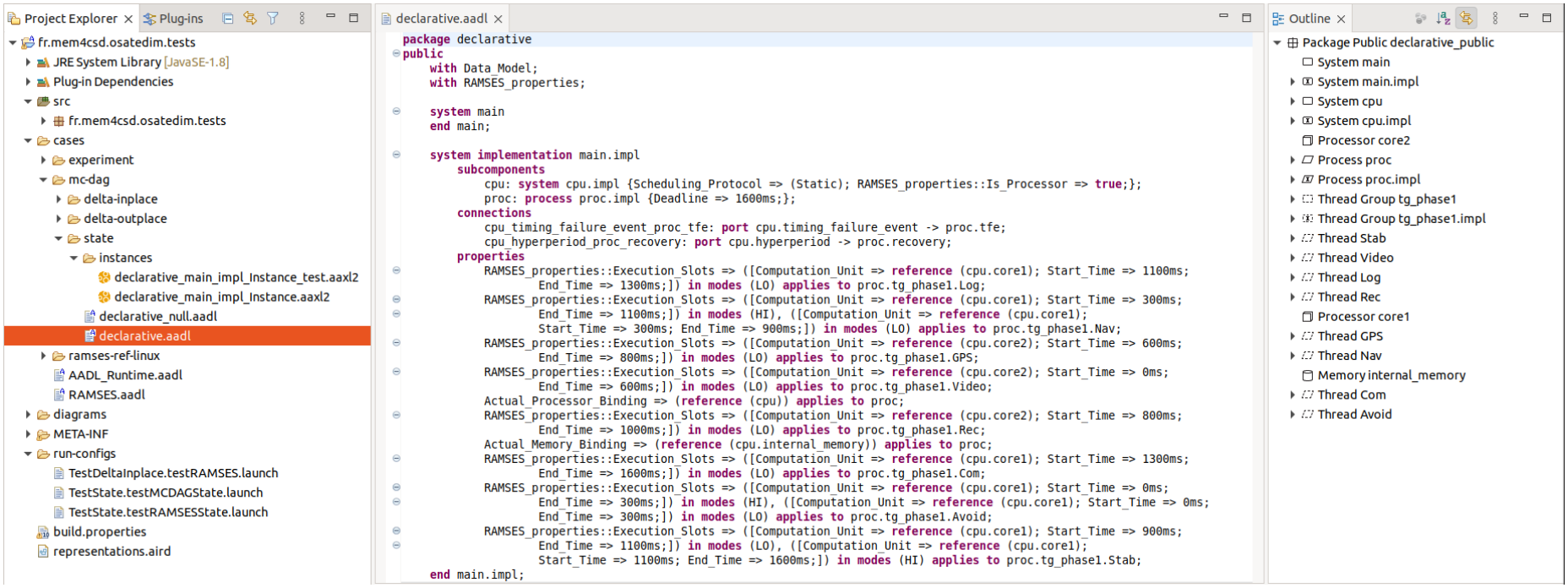
b. The cases/ directory contains the .aadl (Declarative) and .aaxl2 (Instance) models that are used in the case-studies. The models are contained in specific subdirectories according to the case-study (MC-DAG, RAMSES, Experiment) and scenario (state-based, delta-inplace, delta-outplace)
c. To run the tests, go to the ‘Run >> Run Configurations’ menu : In the ‘Run Configurations’ dialog box, you will see the JUnit tests associated with OSATE-DIM



d. Select one of them and Click ‘Run’

Observing the Results

1. After running a particular test, observe the newly created and modified files through the Project Explorer view (in the ‘Plug-in Development’ perspective) or the ‘AADL Navigator’ view (in the AADL perspective).


Project Explorer
Plug-ins
fr.mem4csd.osatedim.tests
JRE System Library [JavaSE-1.8]
Plug-in Dependencies
src
fr.mem4csd.osatedim.tests
cases
experiment
mc-dag
delta-inplace
delta-outplace
state
instances
declarative_main_impl_Instance_test.aaxl2
declarative_main_impl_Instance.aaxl2
declarative_null.aadl
declarative.aadl
ramses-ref-linux
AADL_Runtime.aadl
RAMSES.aadl
diagrams
META-INF
run-configs
TestDeltaInplace.testRAMSES.launch
TestState.testMCDAGState.launch
TestState.testRAMSESState.launch
build.properties
representations.aird
declarative.aadl
package declarative
public
with Data_Model;
with RAMSES_properties;
system main
end main;
system implementation main.impl
subcomponents
cpu: system cpu.impl {Scheduling_Protocol => (Static); RAMSES_properties::Is_Processor => true;};
proc: process proc.impl {Deadline => 1600ms;};
connections
cpu_timing_failure_event_proc_tfe: port cpu.timing_failure_event -> proc.tfe;
cpu_hyperperiod_proc_recovery: port cpu.hyperperiod -> proc.recovery;
properties
RAMSES_properties::Execution_Slots => ([Computation_Unit => reference (cpu.core1); Start_Time => 1100ms;
End_Time => 1300ms;]) in modes (LO) applies to proc.tg_phase1.Log;
RAMSES_properties::Execution_Slots => ([Computation_Unit => reference (cpu.core1); Start_Time => 300ms;
End_Time => 1100ms;]) in modes (HI), ([Computation_Unit => reference (cpu.core1);
Start_Time => 300ms; End_Time => 900ms;]) in modes (LO) applies to proc.tg_phase1.Nav;
RAMSES_properties::Execution_Slots => ([Computation_Unit => reference (cpu.core2); Start_Time => 600ms;
End_Time => 800ms;]) in modes (LO) applies to proc.tg_phase1.GPS;
RAMSES_properties::Execution_Slots => ([Computation_Unit => reference (cpu.core2); Start_Time => 0ms;
End_Time => 600ms;]) in modes (LO) applies to proc.tg_phase1.Video;
Actual_Processor_Binding => (reference (cpu)) applies to proc;
RAMSES_properties::Execution_Slots => ([Computation_Unit => reference (cpu.core2); Start_Time => 800ms;
End_Time => 1000ms;]) in modes (LO) applies to proc.tg_phase1.Rec;
Actual_Memory_Binding => (reference (cpu.internal_memory)) applies to proc;
RAMSES_properties::Execution_Slots => ([Computation_Unit => reference (cpu.core1); Start_Time => 1300ms;
End_Time => 1600ms;]) in modes (LO) applies to proc.tg_phase1.Com;
RAMSES_properties::Execution_Slots => ([Computation_Unit => reference (cpu.core1); Start_Time => 0ms;
End_Time => 300ms;]) in modes (HI), ([Computation_Unit => reference (cpu.core1); Start_Time => 0ms;
End_Time => 300ms;]) in modes (LO) applies to proc.tg_phase1.Avoid;
RAMSES_properties::Execution_Slots => ([Computation_Unit => reference (cpu.core1); Start_Time => 900ms;
End_Time => 1100ms;]) in modes (LO), ([Computation_Unit => reference (cpu.core1);
Start_Time => 1100ms; End_Time => 1600ms;]) in modes (HI) applies to proc.tg_phase1.Stab;
end main.impl;
Outline
Package Public declarative_public
System main
System main.impl
System cpu
System cpu.impl
Processor core2
Process proc
Process proc.impl
Thread Group tg_phase1
Thread Group tg_phase1.impl
Thread Stab
Thread Video
Thread Log
Thread Rec
Processor core1
Thread GPS
Thread Nav
Memory internal_memory
Thread Com
Thread Avoid

2. The tests also generate a comparison report as a test of the correctness of the de-instantiation. This comparison report is generated and reported within the Console view.
3. When de-instantiation takes place, OSATE-DIM also logs information to the console for each performed step.

Running OSATE-DIM Independently

The user may want to test the de-instantiation of their own custom models. In such a case, the user can use the graphical user interface provided by OSATE-DIM to affect de-instantiation. This interface is described in the Tool Demonstration paper, Mittalet al. “OSATE-DIM Solves the Instance Model-View Update Problem in AADL”, provided as extra notes for artifact evaluation.