



SysML to AADL

Pierre Dissaux, AADL committee, Pittsburgh, 16 May 2018

SysML and AADL

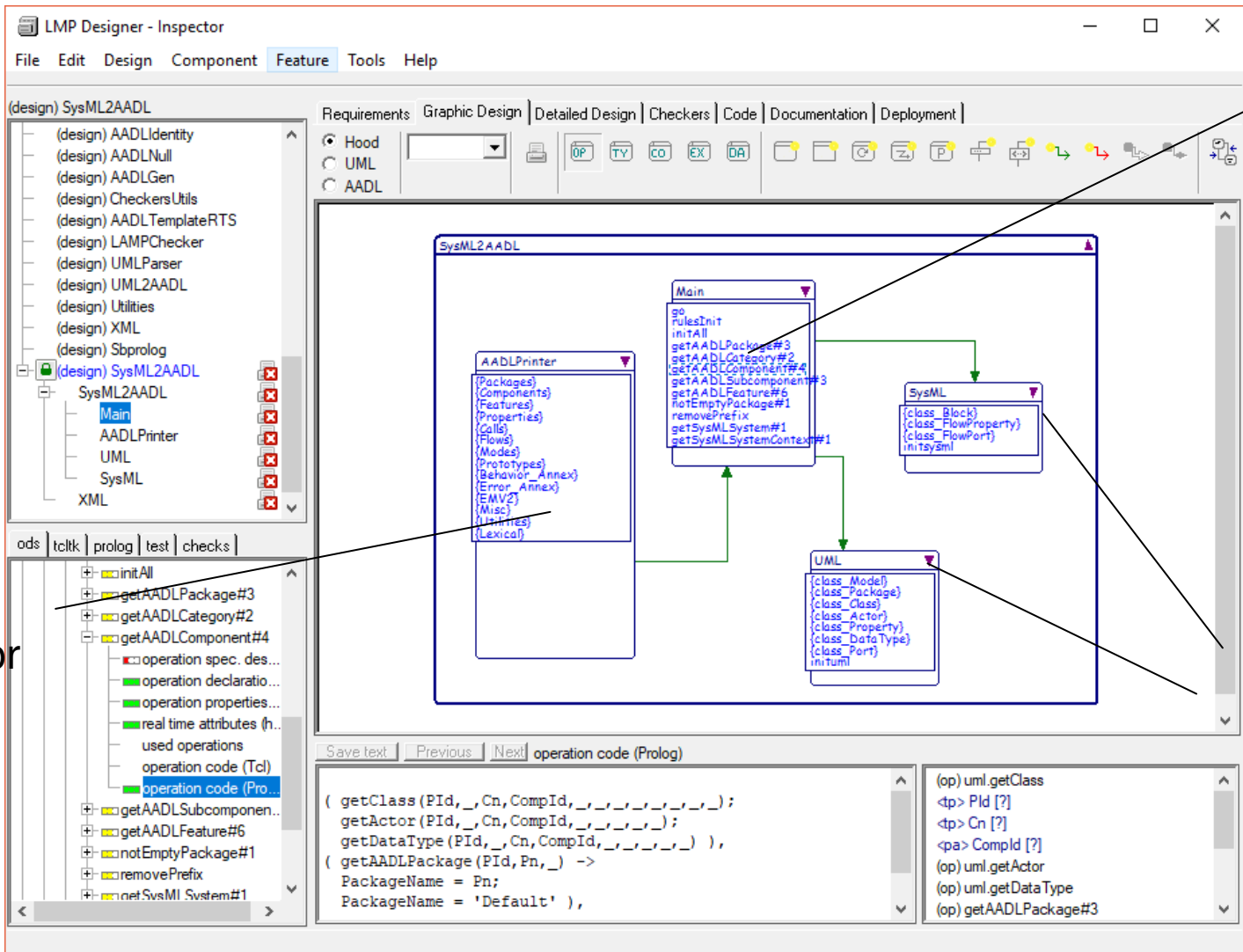
Our return of experience

- Positioning in the workflow/lifecycle:
 - Use of SysML for system engineering (but no unique way to do it)
 - Use of AADL for the refinement of the electronic and software subsystems
 - AADL can also be a way to perform more formal verifications
- Status of the SysML standardisation:
 - Poor tools interoperability (XMI)
 - Complicated by the dependency with core UML
- Converting SysML to AADL:
 - No unique mapping of concepts
 - No unique input format

Our current solution:

- A generic approach to develop custom SysML to AADL transformations
- An illustration of this approach for demonstration purpose:
 - An experimental "import SysML" plugin in AADL Inspector (current version)
 - A toy example inspired from a No Magic case study
- Requires adaptation for real use (mapping + transformation update)

Building the transformation (LMP)



The screenshot shows the LMP Designer - Inspector window. The left pane displays the project structure for SysML2AADL, including components like AADLIdentity, AADLNull, AADLGen, CheckersUtils, AADLTemplateRTS, LAMPChecker, UMLParser, UML2AADL, Utilities, XML, and Sbprolog. The main pane shows a SysML2AADL diagram with components: AADLPrinter, Main, SysML, and UML. The bottom pane shows the operation code (Prolog) for the selected operation, which includes code for getting class, actor, data type, and package information.

AADL generator

Mapping implementation (prolog)

XMI model elements (selection)

operation code (Prolog)

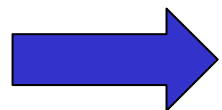
```

    ( getClass (Pid,_,Cn,CompId,_,_,_,_,_,_) ;
      getActor (Pid,_,Cn,CompId,_,_,_,_,_) ;
      getDataType (Pid,_,Cn,CompId,_,_,_,_) ) ,
    ( getAADLPackage (Pid,Pn,_) ->
      PackageName = Pn;
      PackageName = 'Default' ) ,
  
```

operation code (Prolog)

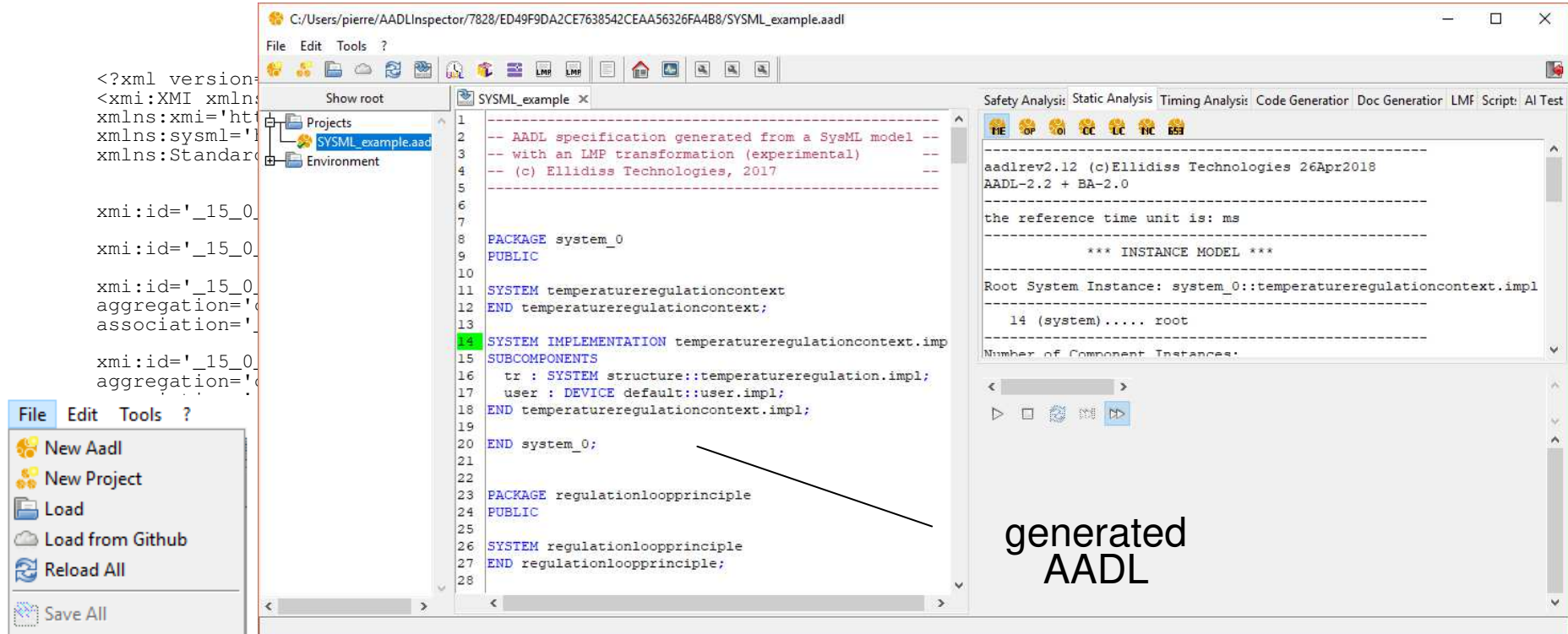
```

    (op) uml.getClass
    <tp> Pid [?]
    <tp> Cn [?]
    <pa> CompId [?]
    (op) uml.getActor
    (op) uml.getDataType
    (op) getAADLPackage#3
  
```



generation of a LMP plugin for AADL Inspector

Using the transformation (AI 1.6)



The screenshot shows the AADLInspector application interface. The main editor displays the generated AADL code, which includes a SysML model transformation. The right panel shows the instance model, including the root system instance and its components. The File menu is open, highlighting the 'Import...' option.

generated AADL

```

1  -- AADL specification generated from a SysML model --
2  -- with an LMP transformation (experimental) --
3  -- (c) Ellidiss Technologies, 2017 --
4
5
6
7
8  PACKAGE system_0
9  PUBLIC
10
11  SYSTEM temperatureregulationcontext
12  END temperatureregulationcontext;
13
14  SYSTEM IMPLEMENTATION temperatureregulationcontext.impl
15  SUBCOMPONENTS
16    tr : SYSTEM structure::temperatureregulation.impl;
17    user : DEVICE default::user.impl;
18  END temperatureregulationcontext.impl;
19
20  END system_0;
21
22
23  PACKAGE regulationloopprinciple
24  PUBLIC
25
26  SYSTEM regulationloopprinciple
27  END regulationloopprinciple;
28
  
```

imported SysML (XMI)

```

25_74014_7363' />
25_74014_7363' />
93_730591_7280' />
93_730591_7280' />
74014_7363' visibility='private'
6641_7006'
7625_145116_7362' />
7625_145116_7362' />
</packagedElement>
<packagedElement xmi:type='uml:Association'
xmi:id='_15_0_24400562_1202448781390_818110_7440'>
  <memberEnd
xmi:idref='_15_0_24400562_1202448781390_761663_7441' />
  <memberEnd
xmi:idref='_15_0_24400562_1202448685625_775431_7296' />
  <ownedEnd xmi:type='uml:Property'
xmi:id='_15_0_24400562_1202448781390_761663_7441' visibility='private'
type='_15_0_1b500480_1202184570171_416641_7006'
association='_15_0_24400562_1202448781390_818110_7440' />
</packagedElement>
</packagedElement>
  
```