AADL Compliance Test Suite
Status Report

Julien Delange, Jérôme Hugues, Alexey Khoroshilov

SAE AS-2C AADL Spring Meeting
Minneapolis, April 2012
AADLv2 Test Suite Goals

- to ensure single interpretation of the specification
- to improve quality of the tools and of the specification
- to make limitations of the tools explicit
- to improve interoperability between the tools
Test Suite Organization

- src/ - collection of test models
- requirements/ - requirements tree
- tools/ - infrastructure scripts
- aadl-qa.pl - CLI
src/ - Collection of Test Models

src/ is a hierarchy of test suites:

- test_suite_1/
- test_suite_2/
  - test_suite_2_1/
  - test_suite_2_2/
  - test_suite_2_3/
- test_suite_3/
Test Suite

- test_suite_1/
  - MANIFEST.TS
- test_suite_2/
  - MANIFEST.TS
  - test_suite_2_1/
    - MANIFEST.TS
  - test_suite_2_2/
    - MANIFEST.TS
  - test_suite_2_3/
    - MANIFEST.TS

MANIFEST.TS
Description=AADL Examples
INCLUDE=common
INCLUDE_FILES=SEI.aadl
Test Case

- test_case/
  - MANIFEST.TC
  - anyfile1.aadl
  - anyfile2.aadl

MANIFEST.TC

REQ = /04-Examples/04_4
EXPECTED_RESULT = VALID
EXPECTED_XXX = YYY
Tool Adapter

- tools/manager/ToolAdapter.pm
- tools/manager/OcarinaAdapter.pm

package OcarinaAdapter;
our @ISA = qw(ToolAdapter); # Inherit ToolAdapter

sub do_run {
    my ($self,$test_case) = @_;  
    my %result = ();
    my $code = system("ocarina -aadlv2 $self->{aadlfiles} >$test_case->{LOGDIR}/log.txt 2>&1");
    $result{RESULT} = ($code eq 0) ? "VALID" : "INVALID";
    $result{LOG} = `cat $test_case->{LOGDIR}/log.txt`;
    return \%result;
}
Verdict

MANIFEST.TC
REQ = /04-Examples/04_4
EXPECTED_RESULT = VALID
EXPECTED_XXX1_OPT = YYY
EXPECTED_XXX2_MATCH=REEXP

RESULT MAP
RESULT = VALID
XXX1 = YYY1
XXX2 = YYY2
XXX3 = YYY3

PASS  FAIL
### Ocarina:/AADL-CTS/01-Syntax

**Problem Summary**

Click on lines in the table to see the details about each problem.

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Severity</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>/package_spec/01-no-pri-no</td>
<td>failed</td>
<td>RESULT is 'INVALID' instead of 'VALID'</td>
</tr>
</tbody>
</table>

Messages from the test:

```
01-no-pri-no.aadl:7:07: parsing Defining_Identifier, identifier 'package_specx' is expected, found token ';
01-no-pri-no.aadl:7:07: parsing AADL_Declaration, unexpected identifier 'package_spec'
```

Reference to the specification: [Requirements/01-Syntax/package_spec/01-no-pri-no](#)
- **01-Syntax**
  - nonterminals
    - positive & negative test purposes
- **02-Semantic**
  - Subrequirements of Naming, Legality and other rules
    - positive & negative test purposes
- **03-Instance**
  - empty
- **04-Examples**
  - Examples from the specification
property expressions can be evaluated to known values, if necessary, by considering all possible runtime states. A given property definition may have a default expression.

11.1 Property Sets

(1) A property set defines a named group of property types, property definitions, and property constant values.

Syntax

property_set ::= 

    defining_property_set_identifier is

    { import_declaration }*
    { property_type_declaration
        | property_definition_declaration
        | property_constant* 
    }

end defining_property_set_identifier;

Naming Rules

(N1) Property set defining identifiers must be unique in the global namespace.

(N2) The defining identifier following the reserved word end must be identical to the defining identifier following the reserved word property set.

(N3) Associated with every property set is a property set namespace that contains the defining identifiers for all

Test Purposes

AADL-11.1-N2

(N2) The defining identifier following the reserved word `end` must be identical to the defining identifier following the reserved word `property set`.

Test purposes:

01-pos

```plaintext
property set MyProp is
  Access_Rights3 : type enumeration (read_only2, write_only2, read_write2);
Pkg_Access_Right1 : MyProp::Access_Rights3 applies to (package);
end MyProp;
```

02-neg

```plaintext
property set MyProp is
  Access_Rights3 : type enumeration (read_only2, write_only2, read_write2);
Pkg_Access_Right1 : MyProp::Access_Rights3 applies to (package);
end MyProp;
```
Generated Staff

- `gensrc/`  -- similar to `src/`, but generated from `requirements/`

- `journals/`  -- plain logs of all runs

- `reports/`  -- result reports
aadl-qa.pl

- `aadl-qa.pl gensrc`
  - requirements/ → gensrc/
- `aadl-qa.pl run <tool_1> <tool_2>`
  - run all the tests
- `aadl-qa.pl clean`
  - remove all generated staff
- `aadl-qa.pl --help`
  - help message
Reference to the Specification

Report for Ocarina run 20120415-235155

Test Info
Tests Started At 15-Apr-2012 23:51:55
Tests Finished At 15-Apr-2012 23:51:57
Operating System Ubuntu 11.10

Tests executed

Ocarina
AADL-CTS Failures: 1 Passed: 17

Ocarina:/AADL-CTS

Problem Summary
Click on lines in the table to see the details about each problem.

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Severity</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>/04-Examples/04_2/Aircraft_Cockpit_4.2</td>
<td>failed</td>
<td>RESULT is 'INVALID' instead of 'VALID'</td>
</tr>
</tbody>
</table>

Messages from the test:

Aircraft_Cockpit_4.2.aadl:17:48: Avionics::DataTypes::AirData (identifier) is not visible
Cannot analyze AADL specifications

Reference to the specification: /Requirements/04-Examples/04_2/Aircraft_Cockpit_4.2
(10) When a component implementation is declared as an extension of another component implementation and it is declared in both the public and private section of a package, then the `extends` is specified with the public section and the extension may include prototype bindings. The component implementation in the private section is considered to complete the declaration in the public section, i.e., its name can be interpreted as reference to the defining name of the component implementation declaration in the public portion.

(11) An `import declaration` specifies which packages and property sets can be named in qualified references to items in other packages or property sets. Packages can initially be declared within an `import declaration` without classifiers to set up an initial collection of package with use restrictions on other packages.

(12) An `alias declaration` introduces local identifiers as short names for long names. It does so for package names and for classifier type references qualified by a package name. The short name may differ from the identifier of the long name to avoid name conflicts.

(13) Property associations declared in the properties section of a package are associated with the package represented by the declaration. Packages with separate public and private package declarations can have different property values for the same property.

**Processing Requirements and Permissions**

(14) A method of implementation is permitted to enforce that the with declaration in a package not be changed to enforce the use restrictions between packages when classifiers are added to the package.

```vhd
package Aircraft::Cockpit
public
 with Avionics::DataTypes, Safety_Properties;
 AirData renames data Avionics::DataTypes::AirData;
 system MFD
 features
 Airdata; in data port AirData;
 properties
 Safety_Properties::Safety_Criticality => high;
 end MFD;
 end Aircraft::Cockpit;
```

### 4.3 Component Types

(1) A component type specifies the external interface of a component that its implementations satisfy. It contains declarations that represent features of a component and property associations. Features of a component are ports, feature groups, required access to externally provided data, subprogram, and bus components, and parameter declarations for the specification of the data values that flow into and outgoing subprograms. The ports and feature groups of a component can be connected to ports or feature groups of other components to represent control and data interaction between components.
property set Safety_Properties is
  CLevels : type enumeration (low, medium, high);
  Safety_Criticality : Safety_Properties::CLevels applies to (system);
end Safety_Properties;

package Avionics::DataTypes
public
  data AirData
end AirData;
end Avionics::DataTypes;

package Aircraft::Cockpit
public
  with Avionics::DataTypes, Safety_Properties;
  AirData renames data Avionics::DataTypes::AirData;
  system MFD
  features
    Airdata: in data port AirData;
  properties
    Safety_Properties::Safety_Criticality => high;
package TypeExample

public

system File System;

features
  -- access to a data component
  root: requires data access file;
end File System;

process Application
features
  -- a data out port
  result: out data port App::result_type;
  home: requires data access FileSystem::Directory::hashed;
end Application;

thread Calculate
prototypes
  -- A data type to be used as type for the input and result port
  data_type: data;
features
  input: in data port data_type;
  result: out data port data_type;
end Calculate;

thread Compute_Distance extends Calculate (data_type => data App::Distance)
end Compute_Distance;
end TypeExample;

Information

Ocarina failed: RESULT is 'INVALID' instead of 'VALID'
TypeExample_4_3.aadl:40:08: input (port spec) points to data_type (entity reference), which is not of an adequate kind
TypeExample_4_3.aadl:40:08: input (port spec) does not point to anything or to something unreachable
TypeExample_4_3.aadl:41:08: result (port spec) points to data_type (entity reference), which is not of an adequate kind
TypeExample_4_3.aadl:41:08: result (port spec) does not point to anything or to something unreachable
Cannot analyze AADL specifications
Test Suite Structure

- Syntax Tests
  - Positive
  - Negative
- Semantic Tests
  - Positive
  - Negative
- Instance Builder Tests
- Misc
  - Examples from the spec
  - Sample models
Conclusions

- Prove of Concept is available
- AADL-QA Open Source Project on TuxFamily.org
  - Infrastructure scripts - GPLv3
  - AADL models – various sources
- Prerequisites:
  - perl + YAML
  - JRE
Open Questions

- Syntax/Semantic Tests
  - Error message/location checks
- Instance Builder Tests
  - Query language
  - Dump of instance
Thank you!

Alexey Khoroshilov
khoroshilov@ispras.ru