AADL Inspector 1.2

Instance Model Overview & Declarative Model Metrics

Ellidiss Technologies
http://www.ellidiss.fr
aadl@ellidiss.fr

1 AADL Metrics

The *metrics* plugin provides a few statistics on the analysed **AADL** specification. It is composed of four parts that are displayed sequentially in the plugin report area of **AADL Inspector**:

- **AADL** Parser Information
- The **AADL** Instance Model Overview
- The **AADL** Instance Model Metrics
- The **AADL** Declarative Model Metrics

AADL Inspector analyses a set of text files containing valid **AADL** specifications that are organised as a set of Packages and Property Sets. This raw set of definitions represents the **AADL** Declarative Model.

However, most analysis tools require operating on the **AADL** Instance Model, e.g. on the instantiated hierarchy of subcomponents starting from an identified Root System Instance belonging to the Declarative Model. The Instance Model must also take into accounts the software binding properties to the hardware. It is thus important for the end user to get an overview of the Instance Model that will be processed by analysis tools such as **Cheddar** or **Marzhin**.

1.1 AADL Parser Information

AADL Inspector uses the *aadlrev* parser to perform the **AADL** syntactic analysis and store the model into an appropriate form for further processing. Depending on the kind of constructs that are recognized in the source **AADL** model, the core language or the various sub-languages defined by **AADL** annexes will be analysed and merged together into a common fact base.

The *metrics* pluging shows which version of the **AADL** core and annexes have been recognized while parsing the source model:

```
--------------------------------------------------
aadlrev2.4 (c)Ellidiss Technologies 20Aug2013
AADL-2.1 + BA-1.1 + EMV2-0.95
--------------------------------------------------
```

1.2 AADL Instance Model Overview

As opposed to most other **AADL** processing tools, **AADL Inspector** does not require the instance model to be built globally and stored separately from the original source declarative model. On the contrary, only the parts of the instance model that are required for a given processing feature are dynamically produced when required. This approach ensures that the instance model is always synchronized with the declarative model.

The first step that is required for building the **AADL** instance model is the identification of the Root System Instance. This entity that is be part of the source declarative model represents the "*System to Design*" and must be a valid **AADL** system implementation classifier.

AADL Inspector automatically recognizes the Root System Instance according to the following criteria in decreasing priority order:

- the first system implementation classifier found that contains an `AI::Root_System` property association.
- the first system implementation classifier found that contains an `Actual_Processor_Binding` property association.
- the first system implementation classifier found that is not instantiated as a subcomponent within the current project, i.e. within the set of loaded **AADL** source files that constitute the input model.

Note that these rules imply that the file loading order may have an impact on the detection of the Root System Instance. It is thus recommended to load an AADL Inspector Configuration file (`.aic`) that lists all the **AADL** files that contribute to a given project and will ensure that they are processed in the right order.

Once the Root System Instance has been recognized, the metrics plugin shows the

hierarchy of processors, processes, threads and subprograms that can be found by following the hardware/software binding properties and the subcomponent instantiation declarations recursively from this starting point. The line number in the source declarative model is given for each instance model entity and corresponds to the related subcomponent statement.

Each instance model entity is identified by its unique instance `id` within the current Root System Instance (`root`). This `id` has the following structure:

```
root
  [ . hardware_component_instance_id
    [ . software_component_instance_id ] ]
```

Here is a fragment of an instance model overview:

```
-----------------------------------------------------
            *** INSTANCE MODEL ***
-----------------------------------------------------
Root System Instance: Display_System::AI_adaptation::display.impl
-----------------------------------------------------
   17 (system)..... root
 9146 (processor)... root.s_cdu_l_pn.cpm (RM)
 6662 (process)..... root.s_cdu_l_software.p_flight_manager
 5688 (thread)..... t_csci_status_1_hz_flight_manager (PERIODIC)
 5689 (thread)..... t_csci_status_5_hz_flight_manager (PERIODIC)
 5690 (thread)..... t_efp_data_5_hz_flight_manager (PERIODIC)
 5691 (thread)..... t_flight_plan_5_hz_flight_manager (PERIODIC)
 9615 (processor)... root.s_li_mfd_pn.cpm (RM)
 7062 (process)..... root.s_li_mfd_software.p_eicas_manager
 5842 (thread)..... t_cdu_disp_cmds_20_hz_eicas_manager (PERIODIC)
 5844 (thread)..... t_efp_data_10_hz_eicas_manager (PERIODIC)
 5845 (thread)..... t_efp_data_20_hz_eicas_manager (PERIODIC)
 7063 (process)..... root.s_li_mfd_software.p_mfd_display_manager
 5978 (thread)..... t_comm_alert_data_10_hz_mfd_display_manager
 7476 (process)..... root.s_ro_pp_software.p_dme_manager
 6305 (thread)..... t_comm_alert_data_10_hz_dme_manager (PERIODIC)
 9615 (processor)... root.s_ri_mfd_pn.cpm (RM)
 7062 (process)..... root.s_ri_mfd_software.p_eicas_manager
 5842 (thread)..... t_cdu_disp_cmds_20_hz_eicas_manager (PERIODIC)
 5844 (thread)..... t_efp_data_10_hz_eicas_manager (PERIODIC)
 5845 (thread)..... t_efp_data_20_hz_eicas_manager (PERIODIC)
```

1.3 AADL Instance Model Metrics

After the complete instance model hierarchy has been displayed, the *metrics plugin* shows the number of processor, process, thread and subprogram subcomponents that has been found.

```
-------------------------------------------------
Number of Component Instances:
-------------------------------------------------
- Processors:  5
- Processes:   22
- Threads:     123
- Subprograms: 0
-------------------------------------------------
```

1.4 AADL Declarative Model Metrics

In a similar way, the number of each kind of **AADL** core and annexes statement that has been detected while parsing the source declarative model is displayed in the report area of **AADL Inspector**:

```
-------------------------------------------------
           *** DECLARATIVE MODEL ***
-------------------------------------------------
Number of Packages:.............................2
Number of Component Types:.....................642
Number of Component Implementations:...........101
Number of Subcomponents:.......................119
Number of Call Sequences:.......................0
Number of Subprogram Calls:.....................0
Number of Features:...........................2423
Number of Connections:........................1484
Number of Property Associations:..............1085
---- Prototypes ---------------------------------
Number of Prototypes:...........................0
Number of Prototype Bindings:...................0
---- Flows --------------------------------------
```

```
Number of Flow Specifications:.............1559
Number of Flow Implementations:............1672
---- Modes ---------------------------------------
Number of Modes:...............................0
Number of Mode Transitions:....................0
---- Properties ----------------------------------
Number of Property Sets:.......................2
Number of Property Types:......................0
Number of Property Definitions:...............13
Number of Property Constants:..................0
---- Annexes -------------------------------------
Number of Annexes:.............................0
---- Behavior Annex Items ------------------------
Number of Behavior Annex Variables:............0
Number of Behavior Annex States:...............0
Number of Behavior Annex Transitions:..........0
Number of Behavior Annex Conditions:...........0
Number of Behavior Annex Actions:..............0
---- Error Annex v1 Items ------------------------
Number of Error Annex Properties:..............0
---- Error Annex v2 Items ------------------------
Number of Error Type Definitions:..............0
Number of Error Type Set Definitions:..........0
Number of Error Type Mappings:.................0
Number of Error Type Transformations:..........0
Number of Error Behavior Definitions:..........0
Number of Error Event Definitions:.............0
Number of Error Behavior States:...............0
Number of Error Behavior Transitions:..........0
Number of Error Propagation Definitions:.......0
Number of Error Source Definitions:............0
Number of Error Sink Definitions:..............0
Number of Error Path Definitions:..............0
Number of Outgoing Error Propagations:.........0
Number of Error Detections:....................0
Number of Composite Error States:..............0
Number of Connection Error Sources:............0
Number of Propagation Point Definitions:.......0
Number of Propagation Point Connections:.......0
-------------------------------------------------
```

www.ellidiss.com

Sales office:
TNI Europe Limited
Triad House
Mountbatten Court
Worall Street
Congleton
Cheshire
CW12 1AG
UK
info@ellidiss.com
+44 1260 291 449

Technical support :
Ellidiss Technologies
24 quai de la douane
29200 Brest
Brittany
France

aadl@ellidiss.fr
+33 298 451 870