AADL Inspector 1.2
AADL Legality Rules

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

1 Static Rules Checkers

AADL Inspector comes with a set of pre-defined static rules checkers. Their goal is to cover the various semantic rules specified by the **AADL** standard. In the standard document **SAE-AS5506B**, they are classified into three categories:

- *naming rules*: specify the appropriate usage of identifiers.
- *legality rules*: express semantic restrictions on syntactically valid text.
- *consistency rules*: define constraints related to the instance model.

This document describes the **AADL Inspector** Legality Rules checker.

These standard legality rules are not fully implemented in the current version of the tool and the following sub-sections provide details about the actual **AADL** rules coverage. For easier traceability, precise references to the standard document are given for each implemented rule.

2 AADL subset

Legality rule identifiers are composed the letter `L` followed by an integer number. Numbering restarts at the beginning of each section in the **AADL** standard document. Rules that are currently implemented refer to the following sections of the document **SAE-AS5506B**:

- *Packages*: section 4.2
- *Component Types*: section 4.3
- *Component Implementations*: section 4.4

2.1 Packages (AS5506B-4.2)

Checked legality rules for **AADL** packages are `L1`, `L2` and `L3`.

2.2 Component Types (AS5506B-4.3)

Checked legality rule for **AADL** component types is `L1`, `L3`, `L4` and `L6`. The other rules will be implemented in the next releases of the product.

2.3 Component Implementations (AS5506B-4.4)

Checked legality rule for **AADL** component implementations is `L1`, `L3`, `L4`, `L5`, `L6`, `L7`, `L8` and `L9`. The other rules will be implemented in the next releases of the product.

3 Test Cases

3.1 Packages (AS5506B-4.2)

L1 The defining package name following the reserved word **end** must be identical to the defining package name following the reserved word **package**.

```
PACKAGE package1
PUBLIC

END package2;
```

WARNING line 9: begin and end identifiers should match for Package package1 (ref. AADL v2 - 4.2 - L1)

L2 For each package there may be at most one **public** section declaration and one **private** section declaration. These two sections may be declared in a single package declaration or in two separate package declarations.

Test case 1:

```
PACKAGE package1
PUBLIC

END package1;

PACKAGE package1
PUBLIC

END package1;
```

WARNING lines 7 and 12: there should be only one Public section for Package package1 (ref. AADL v2 - 4.2 - L2)

Test case 2:

```
PACKAGE package1
PRIVATE

END package1;

PACKAGE package1
PRIVATE

END package1;
```

WARNING lines 12 and 7: there should be only one Private section for Package package1 (ref. AADL v2 - 4.2 - L2)

L3 A component implementation may be declared in both the public and private part of a package. In that case the declaration in the public part may only contain a **properties** subclause and a **modes** subclause.

```
PACKAGE P1
PUBLIC

THREAD T1 END T1;

THREAD T2 END T2;

THREAD IMPLEMENTATION T2.I
END T2.I;

PROCESS P1
END P1;

PROCESS IMPLEMENTATION P1.I
SUBCOMPONENTS
  X : THREAD T2.I;
PROPERTIES
  Source_Code_Size => 100 KBytes;
END P1.I;
```

```
PRIVATE

THREAD IMPLEMENTATION T1.I
END T1.I;

PROCESS IMPLEMENTATION P1.I
SUBCOMPONENTS
  Y : THREAD T1.I;
END P1.I;

END P1;
```

ERROR line 21: There should be only Properties and Modes subclauses in the public part of component implementation P1.I (ref. AADL v2 - 4.2 - L3)

L4 The component category in an alias declaration must match the category of the referenced component type.

```
PACKAGE P1
PUBLIC

PROCESS T
END T;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

T RENAMES THREAD P1::T;

END P2;
```

ERROR lines 10 and 18 Component categories do not match (ref. AADL v2 - 4.2 - L4)

3.2 Component types (AS5506B-4.3)

L1 The defining identifier following the reserved word **end** must be identical to the defining identifier that appears after the component category reserved word.

```
PACKAGE P1
PUBLIC

THREAD T1
END T;

END P1;
```

WARNING line 10: begin and end identifiers should match for Component Type T1 (ref. AADL v2 - 4.3 - L1)

L2 The **prototypes**, **features**, **flows**, **modes**, and **properties** subclauses are optional. If a subclause is present but empty, then the reserved word **none** followed by a semi-colon must be present.

This rule is checked by the ***AADL*** *syntactic analyser (*aadlrev*).*

L3 The category of the component type being extended must match the category of the extending component type, i.e., they must be identical or the category being extended must be **abstract**.

```
PACKAGE P1
PUBLIC

THREAD TH1
END TH1;

PROCESS PR1 EXTENDS TH1
END PR1;

END P1;
```

ERROR lines 13 and 10: the category of PR1 must match the category of TH1 (ref. AADL v2 - 4.3 - L3)

L4 The classifier being extended in a component type extension may include prototype bindings. There must be at most one prototype binding for each prototype, i.e., once bound a prototype binding cannot be overwritten by a new binding in a component type extension.

```
PACKAGE P1
PUBLIC

DATA D
END D;

SUBPROGRAM S
END S;

THREAD T1
PROTOTYPES
  p1 : DATA;
  p2 : SUBPROGRAM;
END T1;

THREAD T2 EXTENDS T1
( p1 => DATA D,
  p2 => SUBPROGRAM S)
END T2;

THREAD T3 EXTENDS T2
( p1 => DATA D )
END T3;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

T4 RENAMES THREAD P1::T2;
D RENAMES DATA P1::D;
S RENAMES SUBPROGRAM P1::S;

THREAD T5 EXTENDS T4
END T5;

THREAD T6 EXTENDS T5
( p1 => DATA D,
  p2 => SUBPROGRAM S )
```

```
END T6;

END P2;
```

ERROR lines 27 and 22: prototype p1 is already bound in component type T2 (ref. AADL v2 - 4.3 - L4)
ERROR lines 44 and 22: prototype p1 is already bound in component type T2 (ref. AADL v2 - 4.3 - L4)
ERROR lines 45 and 23: prototype p2 is already bound in component type T2 (ref. AADL v2 - 4.3 - L4)

L5 A component type must not contain both a `requires_modes_subclause` and a `modes_subclause`.

*This rule is checked by the **AADL** syntactic analyser (*aadlrev*).*

L6 If the extended component type and an ancestor component type in the extends hierachy contain modes subclauses, they must both be `requires_modes_subclause` or `modes_subclause`.

```
PACKAGE P1
PUBLIC

SYSTEM S1
MODES
  M1 : INITIAL MODE;
  M2 : MODE;
END S1;

SYSTEM S2 EXTENDS S1
REQUIRES MODES
  M3 : MODE;
  M4 : MODE;
END S2;

END P1;

PACKAGE P2
PUBLIC
WITH P1;
```

```
S RENAMES SYSTEM P1::S2;

SYSTEM S3 EXTENDS S
REQUIRES MODES
  M5 : MODE;
END S3;

END P2;
```

ERROR lines 17 and 11: component type S2 and component type S1 should have both modes or requires modes subclauses (ref. AADL v2 - 4.3 - L6)
ERROR lines 31 and 11: component type S3 and component type S1 should have both modes or requires modes subclauses (ref. AADL v2 - 4.3 - L6)

3.3 Component implementations (AS5506B-4.4)

L1 The pair of identifiers separated by a dot (".") following the reserved word **end** must be identical to the pair of identifiers following the reserved word **implementation**.

Test case 1:

```
PACKAGE P1
PUBLIC

PROCESS P1
END P1;

PROCESS IMPLEMENTATION P1.I
END P1.J;

PROCESS P2
END P2;

PROCESS IMPLEMENTATION P2.I
END P1.I;

END P1;
```

WARNING line 13: begin and end identifiers should match for Component Implementation P1.I (ref. AADL v2 - 4.4 - L1)

WARNING line 19: begin and end identifiers should match for Component Implementation P2.I (ref. AADL v2 - 4.4 - L1)

L2 The **prototypes**, **subcomponents**, **connections**, **calls**, **flows**, **modes**, and **properties** subclauses are optional. If they are present and the set of feature or required subcomponent declarations or property associations is empty, none followed by a semi-colon must be present in that subclause.

*This rule is checked by the **AADL** syntactic analyser (*aadlrev*).*

L3 The category of the component implementation must be identical to the category of the component type for which the component implementation is declared.

```
PACKAGE P1
PUBLIC

THREAD T
END T;

SUBPROGRAM IMPLEMENTATION T.I
END T.I;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

T RENAMES THREAD P1::T;

SUBPROGRAM IMPLEMENTATION T.J
END T.J;

END P2;
```

ERROR lines 10 and 13: component categories must be identical for component type T and its implementation T.I (ref. AADL v2 - 4.4 - L3)
ERROR lines 21 and 24: component categories must be identical for component type T and its implementation T.J (ref. AADL v2 - 4.4 - L3)

L4 If the component implementation extends another component implementation, the category of both must match, i.e., they must be identical or the category being extended must be **abstract**.

```
PACKAGE P1
PUBLIC

ABSTRACT D
END D;

PROCESS E
END E;

SYSTEM F
END F;

ABSTRACT IMPLEMENTATION D.I
END D.I;

PROCESS IMPLEMENTATION E.I EXTENDS D.I
END E.I;

SYSTEM IMPLEMENTATION F.J EXTENDS E.I
END F.J;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

H RENAMES SYSTEM P1::F;

DATA G
END G;

DATA IMPLEMENTATION G.K EXTENDS H.J
END G.K;

END P2;
```

ERROR lines 21 and 24: component categories must be identical for component

implementations F.J and E.I (ref. AADL v2 - 4.4 - L4)
ERROR lines 24 and 38: component categories must be identical for component implementations G.K and F.J (ref. AADL v2 - 4.4 - L4)

L5 The classifier being extended in a component implementation extension may include prototype bindings. There must be at most one prototype binding for each unbound prototype.

```
PACKAGE P1
PUBLIC

DATA D END D;
SUBPROGRAM S END S;
THREAD T1 END T1;
THREAD T2 EXTENDS T1 END T2;
THREAD T3 EXTENDS T2 END T3;

THREAD IMPLEMENTATION T1.i
PROTOTYPES
  p1 : DATA;
  p2 : SUBPROGRAM;
END T1.i;

THREAD IMPLEMENTATION T2.i EXTENDS T1.i
( p1 => DATA D,
  p2 => SUBPROGRAM S)
END T2.i;

THREAD IMPLEMENTATION T3.i EXTENDS T2.i
( p1 => DATA D )
END T3.i;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

T4 RENAMES THREAD P1::T2;
D RENAMES DATA P1::D;
S RENAMES SUBPROGRAM P1::S;
```

```
THREAD T5 EXTENDS T4 END T5;

THREAD IMPLEMENTATION T5.i EXTENDS T4.i
END T5.i;

THREAD T6 EXTENDS T5 END T6;

THREAD IMPLEMENTATION T6.i EXTENDS T5.i
( p1 => DATA D,
  p2 => SUBPROGRAM S )
END T6.i;

END P2;
```

ERROR lines 27 and 22: prototype p1 is already bound in component implementation T2.i (ref. AADL v2 - 4.4 - L5)
ERROR lines 48 and 22: prototype p1 is already bound in component implementation T2.i (ref. AADL v2 - 4.4 - L5)
ERROR lines 49 and 23: prototype p2 is already bound in component implementation T2.i (ref. AADL v2 - 4.4 - L5)

L6 If the component type of the component implementation contains a `requires_modes_subclause` then the component implementation must not contain any modes subclause.

```
PACKAGE P1
PUBLIC

SYSTEM S1
FEATURES
  E : IN EVENT PORT;
REQUIRES MODES
  M0 : INITIAL MODE;
END S1;

SYSTEM IMPLEMENTATION S1.I
MODES
  M1 : MODE;
  T1 : M0 -[ E ]-> M1;
END S1.I;

SYSTEM S2 EXTENDS S1
```

```
END S2;

SYSTEM IMPLEMENTATION S2.I
MODES
  M2 : MODE;
  T2 : M0 -[ E ]-> M2;
END S2.I;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

S RENAMES SYSTEM P1::S2;

SYSTEM S3 EXTENDS S
END S3;

SYSTEM IMPLEMENTATION S3.I
MODES
  M3 : MODE;
  T3 : M0 -[ E ]-> M3;
END S3.I;

END P2;
```

ERROR lines 18 and 13: component implementation S1.I must not contain any modes subclause (ref. AADL v2 - 4.4 - L6)
ERROR lines 27 and 13: component implementation S2.I must not contain any modes subclause (ref. AADL v2 - 4.4 - L6)
ERROR lines 44 and 13: component implementation S3.I must not contain any modes subclause (ref. AADL v2 - 4.4 - L6)

L7 If modes are declared in the component type, then modes cannot be declared in component implementations.

```
PACKAGE P1
PUBLIC

SYSTEM S1
FEATURES
```

```
  E : IN EVENT PORT;
MODES
  M0 : INITIAL MODE;
  M1 : MODE;
  T1 : M0 -[ E ]-> M1;
END S1;

SYSTEM IMPLEMENTATION S1.I
MODES
  M2 : MODE;
  T2 : M0 -[ E ]-> M2;
END S1.I;

SYSTEM S2 EXTENDS S1
END S2;

SYSTEM IMPLEMENTATION S2.I EXTENDS S1.I
MODES
  M3 : MODE;
  T3 : M0 -[ E ]-> M3;
END S2.I;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

S RENAMES SYSTEM P1::S2;

SYSTEM S3 EXTENDS S
END S3;

SYSTEM IMPLEMENTATION S3.I EXTENDS S.I
MODES
  M4 : MODE;
  T4 : M0 -[ E ]-> M4;
END S3.I;

END P2;
```

ERROR lines 13 and 20 modes cannot be declared in both component type S1 and component implementation S1.I (ref. AADL v2 - 4.4 - L7)

ERROR lines 13 and 29 modes cannot be declared in both component type S2 and component implementation S2.I (ref. AADL v2 - 4.4 - L7)
ERROR lines 13 and 46 modes cannot be declared in both component type S3 and component implementation S3.I (ref. AADL v2 - 4.4 - L7)

L8 If modes or mode transitions are declared in the component type, then mode transitions can be added in the component implementation. These mode transitions may refer to event or event data ports of the component type and of subcomponents.

This rule is not checked.

L9 The category of a subcomponent being refined must match the category of the refining subcomponent declaration, i.e., they must be identical or the category being refined must be **abstract**.

```
PACKAGE P1
PUBLIC

DATA D END D;
THREAD T END T;
ABSTRACT A END A;
SYSTEM S1 END S1;

SYSTEM IMPLEMENTATION S1.I
SUBCOMPONENTS
  X : DATA D;
  Y : ABSTRACT A;
END S1.I;

DATA IMPLEMENTATION D.I
END D.I;

THREAD IMPLEMENTATION T.I
END T.I;

SYSTEM IMPLEMENTATION S1.J EXTENDS S1.I
SUBCOMPONENTS
  X : REFINED TO THREAD T.I;
  Y : REFINED TO THREAD T;
```

```
END S1.J;

SYSTEM S2 EXTENDS S1
END S2;

SYSTEM IMPLEMENTATION S2.I EXTENDS S1.I
SUBCOMPONENTS
  X : REFINED TO THREAD T.I;
  Y : REFINED TO THREAD T;
END S2.I;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

RENAMES P1::ALL;

SYSTEM IMPLEMENTATION S1.K EXTENDS S1.I
SUBCOMPONENTS
  X : REFINED TO THREAD T.I;
  Y : REFINED TO THREAD T;
END S1.K;

SYSTEM S3 EXTENDS S1
END S3;

SYSTEM IMPLEMENTATION S3.I EXTENDS S1.I
SUBCOMPONENTS
  X : REFINED TO THREAD T.I;
  Y : REFINED TO THREAD T;
END S3.I;

END P2;
```

ERROR lines 28 and 16 Categories do not match (ref. AADL v2 - 4.4 - L9)
ERROR lines 37 and 16 Categories do not match (ref. AADL v2 - 4.4 - L9)
ERROR lines 51 and 16 Categories do not match (ref. AADL v2 - 4.4 - L9)
ERROR lines 60 and 16 Categories do not match (ref. AADL v2 - 4.4 - L9)

L10 For all other refinement declarations the categories must match (see the respective sections).

This rule is not checked.

L11 Component implementations and component implementation extensions must not refine prototypes declared in a component type.

```
PACKAGE P1
PUBLIC

DATA D END D;
SUBPROGRAM S END S;
THREAD T2 EXTENDS T1 END T2;
THREAD T3 END T3;

THREAD T1
PROTOTYPES
  p1 : DATA;
END T1;

THREAD IMPLEMENTATION T1.I
PROTOTYPES
  p2 : SUBPROGRAM;
END T1.I;

THREAD IMPLEMENTATION T2.I EXTENDS T1.I
PROTOTYPES
  p1 : REFINED TO DATA D;
  p2 : REFINED TO SUBPROGRAM S;
END T2.I;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

T4 RENAMES THREAD P1::T2;
D RENAMES DATA P1::D;
S RENAMES SUBPROGRAM P1::S;

THREAD T5 EXTENDS T4 END T5;

THREAD IMPLEMENTATION T5.I EXTENDS T4.I
```

```
END T5.I;

THREAD T6 EXTENDS T5 END T6;

THREAD IMPLEMENTATION T6.I EXTENDS T5.I
PROTOTYPES
  p1 : REFINED TO DATA D;
  p2 : REFINED TO SUBPROGRAM S;
END T6.I;

END P2;
```

ERROR lines 26 and 16: prototype p1 cannot be refined in component implementation T2.I (ref. AADL v2 - 4.4 - L11)
ERROR lines 49 and 16: prototype p1 cannot be refined in component implementation T6.I (ref. AADL v2 - 4.4 - L11)

Ellidiss
Technologies

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