

AADL Inspector 1.9

Quick Start Guide

```
5 WITH deadlock;
6 END deadlock;
7
8 DATA D
9 -- deadlock occurs if concurrency control protocol is removed
10 PROPERTIES
11 Concurrency_Control_Protocol => Priority_Ceiling_Protocol;
12 END D;
13
14 SYSTEMS
15 SUBCOMPONENTS
16 Processor C;
17 P1 : PROCESS P.I;
18 PROPERTIES
19 P1.Processor_Binding => (P1 (cpu1)) applies to proce
20 P1.deadlock.others;
21
22 PROCESS P
23 PROCESS C
24 So... nic_Protocol);
25 END C;
26 Protocol => (P
27 PROCESS P
28 END P;
29
30 PROCESS IMPLEMENTATION P.I
31 SUBCOMPONENTS
32 t1 : THREAD T.I;
33 t2 : THREAD T.I;
```



*Strengthened
by LAMP*

Overview

Project
Browser

AADL Textual
Editor

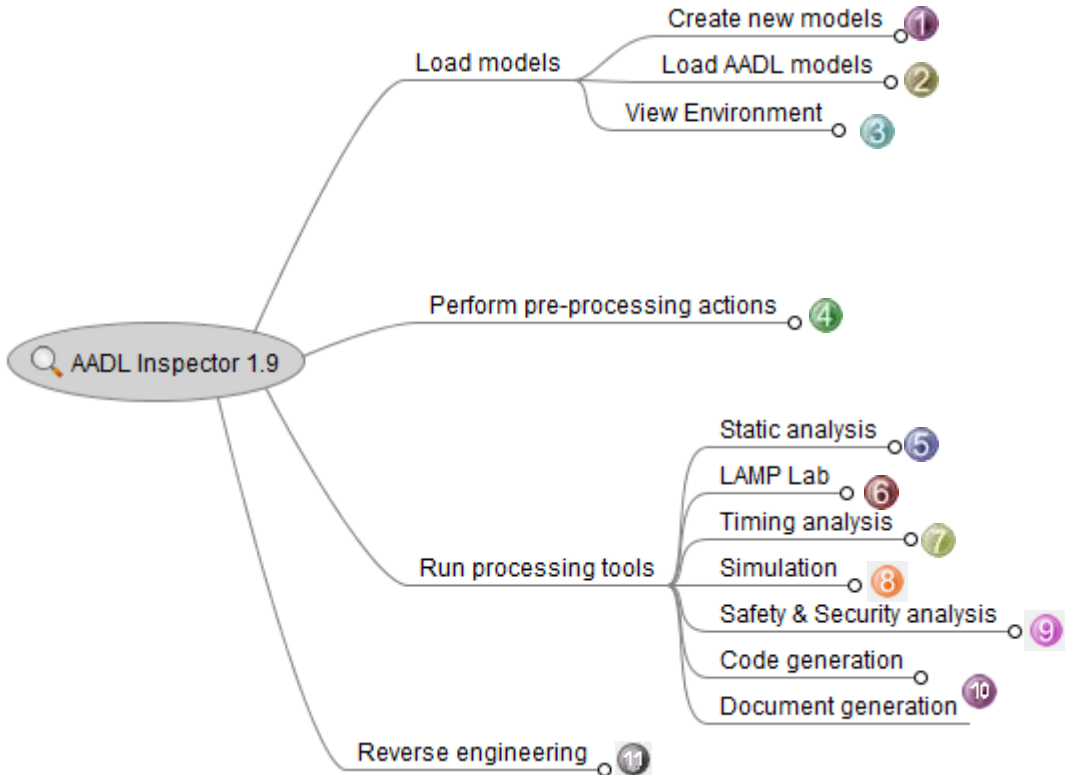
Analysis
Tools

The screenshot displays the AADL Inspector 1.9 interface, which is divided into several main sections:

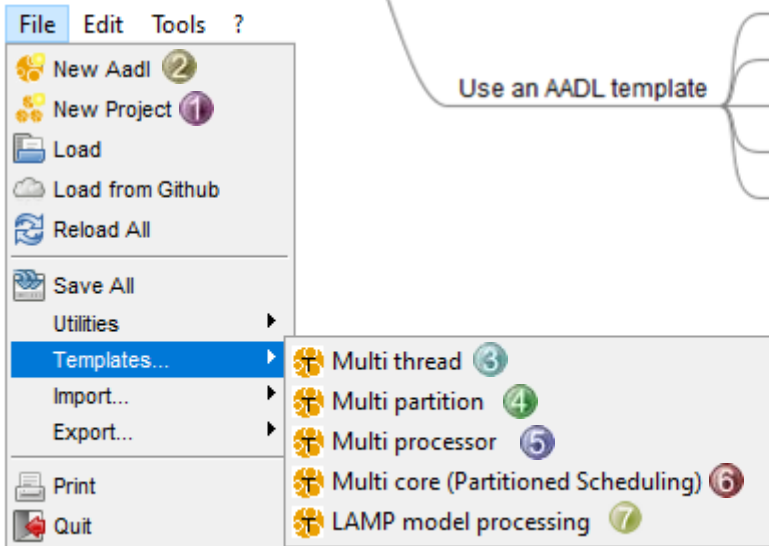
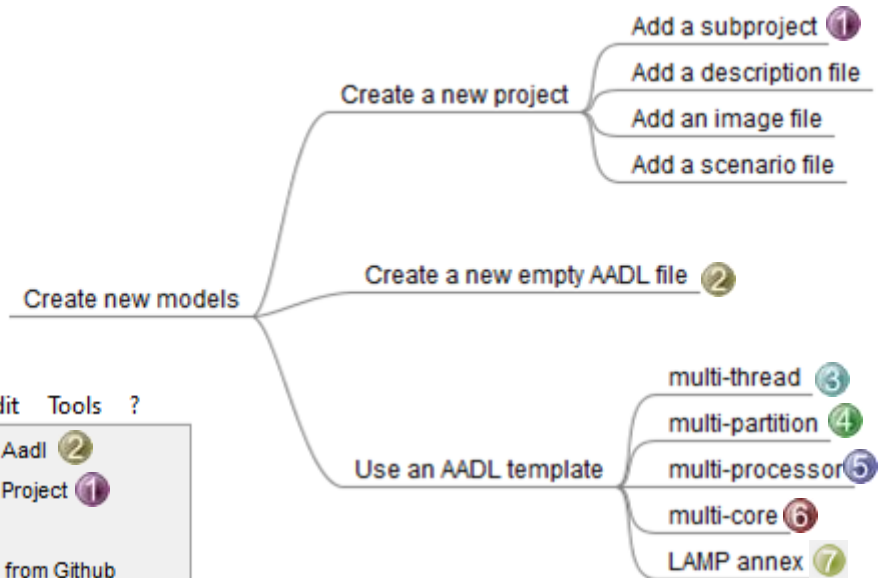
- Project Browser (Left):** Shows a hierarchical tree of project files and folders, including 'control_system.aadl' and various sub-components like 'sensors', 'actuators', and 'network'.
- AADL Textual Editor (Middle-Left):** Displays the AADL code for 'control_system'. The code includes package declarations, system types, and implementation details for sensors, actuators, and network components.
- Analysis Tools (Middle-Right):** Shows a table of analysis results for various components. The table includes columns for Deadline, Computed, Max Checker, Max Marchin, Avg Checkd, Avg Marchin, Min Checkd, and Min Marchin.
- Simulator (Bottom):** Displays a timeline of events and a gauge. The gauge shows a value of approximately 25, with a scale from 0 to 100. The timeline shows various events occurring over time, with a blue arrow pointing to a specific event.

Simulator

Overview



1. Create New Models



1. Create New Models example

The screenshot illustrates the process of creating a new model in AADL Inspector. The 'File' menu is open, showing options like 'New Aadl', 'New Project', 'Load', 'Load from Github', 'Reload All', 'Save All', 'Utilities', 'Templates...', 'Import...', and 'Export...'. The 'Templates...' option is selected, and a sub-menu is displayed with the following options: 'Multi thread', 'Multi partition', 'Multi processor', 'Multi core (Partitioned Scheduling)', and 'LAMP model processing'. A light blue arrow points from the 'Multi thread' option to a 'Set default values' dialog box. The dialog box has a title bar with a gear icon and a close button. It contains two input fields: 'Model Name' with the value 'new_model' and 'Number of Threads' with the value '4'. Below the fields are 'Ok' and 'Cancel' buttons. A light blue arrow points from the 'Ok' button to the code editor. The code editor shows the following AADL code:

```
1  -----  
2  -- AADL Inspector Template  
3  -- Multi Thread Real Time System  
4  -----  
5  
6  PACKAGE new_model_pkg  
7  PUBLIC  
8  WITH ellidiss::math::int;  
9  RENAMES ellidiss::math::int::ALL;  
10  
11 -- Root system interface  
12 SYSTEM new_model  
13 END new_model;  
14  
15 -- Root system body  
16 SYSTEM IMPLEMENTATION new_model.i  
17 SUBCOMPONENTS  
18   hw : PROCESSOR hw.i;  
19   sw : PROCESS sw.i;
```

2. Load Existing Models

The image shows the 'File' menu of the AADL Inspector software. The menu items are: New Aadl, New Project, Load (2), Load from Github (3), Reload All, Save All, Utilities, Templates..., Import... (4), Export..., Print, and Quit. The 'Import...' option is expanded, showing sub-options: Import SysML model (.sysml, .xmi, .model) (4), Import FACE model (.face) (5), Import CAPELLA PA model (.capella) (6), Import Table facts (.csv) (7), Import Textual facts (.pro) (8), and Import Binary facts (.sbp) (9). Annotations include: 1. 'Load examples/all_examples.aic' pointing to the 'Utilities' menu item. 2. 'Load files' with sub-branches 'file navigator' and 'drag and drop' pointing to the 'Load' menu item. 3. 'Load from AADL github' pointing to the 'Load from Github' menu item. 4. 'Import and convert foreign models:' pointing to the 'Import...' menu item. A legend on the right lists the model types corresponding to the numbered icons: 4. SysML (LAMP), 5. FACE (LAMP), 6. Capella P.A. (LAMP), 7. CSV (LAMP), 8. Textual prolog facts, 9. Binary prolog facts.

1 Load examples/all_examples.aic

2 Load files

- file navigator
- drag and drop

3 Load from AADL github

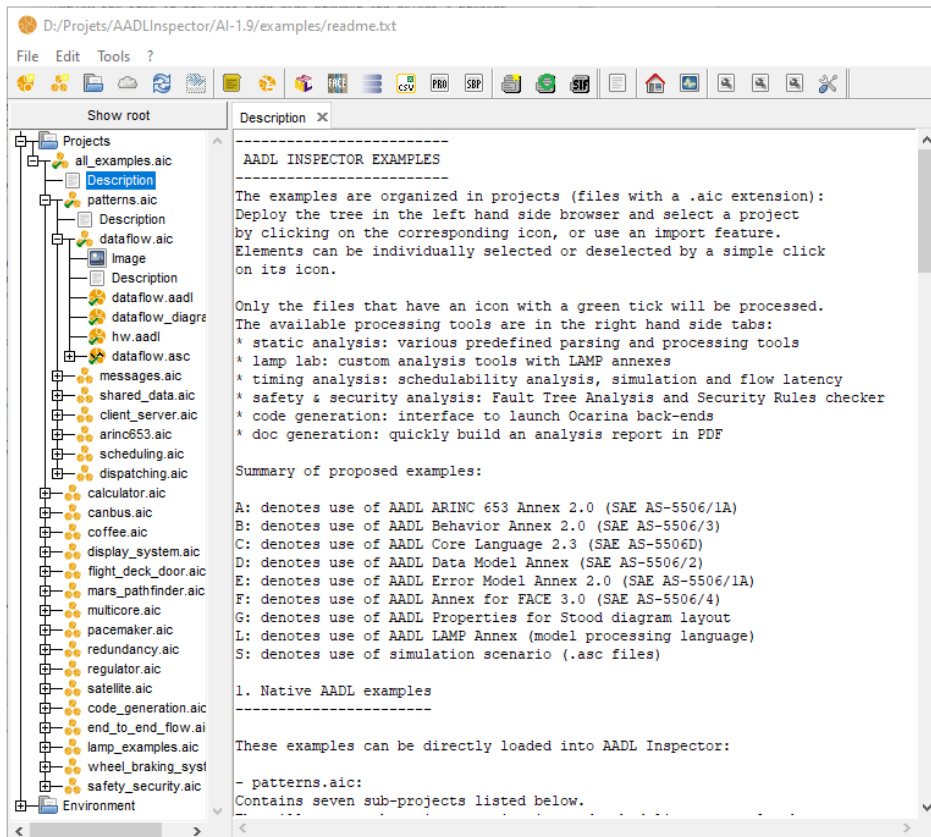
Import and convert foreign models:

- 4 SysML (LAMP)
- 5 FACE (LAMP)
- 6 Capella P.A. (LAMP)
- 7 CSV (LAMP)
- 8 Textual prolog facts
- 9 Binary prolog facts

File Edit Tools ?

- New Aadl
- New Project
- Load 2
- Load from Github 3
- Reload All
- Save All
- Utilities
- Templates...
- Import... 4
 - Import SysML model (.sysml, .xmi, .model) 4
 - Import FACE model (.face) 5
 - Import CAPELLA PA model (.capella) 6
 - Import Table facts (.csv) 7
 - Import Textual facts (.pro) 8
 - Import Binary facts (.sbp) 9
- Export...
- Print
- Quit

2. Load Existing Models example



The screenshot shows the AADL Inspector interface. On the left is a tree view of projects under 'Projects'. The tree includes sub-projects like 'all_examples.aic', 'patterns.aic', 'dataflow.aic', 'dataflow.aadl', 'dataflow_diagram.aic', 'hw.aadl', 'dataflow.asc', 'messages.aic', 'shared_data.aic', 'client_server.aic', 'arinc653.aic', 'scheduling.aic', 'dispatching.aic', 'calculator.aic', 'canbus.aic', 'coffee.aic', 'display_system.aic', 'flight_deck_door.aic', 'mars_pathfinder.aic', 'multicore.aic', 'pacemaker.aic', 'redundancy.aic', 'regulator.aic', 'satellite.aic', 'code_generation.aic', 'end_to_end_flow.ai', 'lamp_examples.aic', 'wheel_braking_sys', and 'safety_security.aic'. The 'Environment' folder is also visible at the bottom of the tree.

The right pane, titled 'Description X', contains the following text:

AADL INSPECTOR EXAMPLES

The examples are organized in projects (files with a .aic extension):
Deploy the tree in the left hand side browser and select a project by clicking on the corresponding icon, or use an import feature.
Elements can be individually selected or deselected by a simple click on its icon.

Only the files that have an icon with a green tick will be processed.
The available processing tools are in the right hand side tabs:
* static analysis: various predefined parsing and processing tools
* lamp lab: custom analysis tools with LAMP annexes
* timing analysis: schedulability analysis, simulation and flow latency
* safety & security analysis: Fault Tree Analysis and Security Rules checker
* code generation: interface to launch Ocarina back-ends
* doc generation: quickly build an analysis report in PDF

Summary of proposed examples:

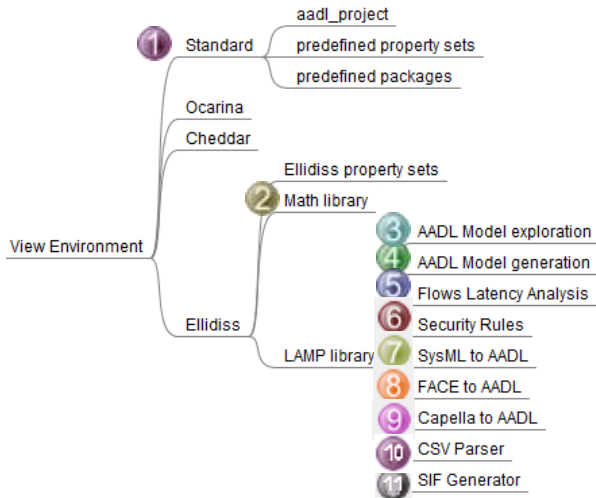
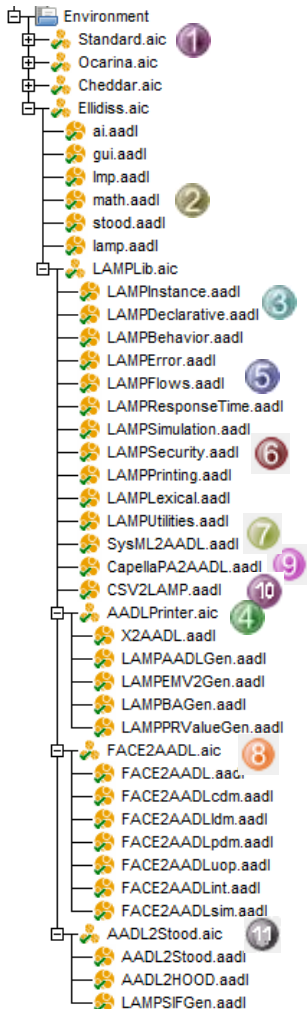
A: denotes use of AADL ARINC 653 Annex 2.0 (SAE AS-5506/1A)
B: denotes use of AADL Behavior Annex 2.0 (SAE AS-5506/3)
C: denotes use of AADL Core Language 2.3 (SAE AS-5506D)
D: denotes use of AADL Data Model Annex (SAE AS-5506/2)
E: denotes use of AADL Error Model Annex 2.0 (SAE AS-5506/1A)
F: denotes use of AADL Annex for FACE 3.0 (SAE AS-5506/4)
G: denotes use of AADL Properties for Stood diagram layout
L: denotes use of AADL LAMP Annex (model processing language)
S: denotes use of simulation scenario (.asc files)

1. Native AADL examples

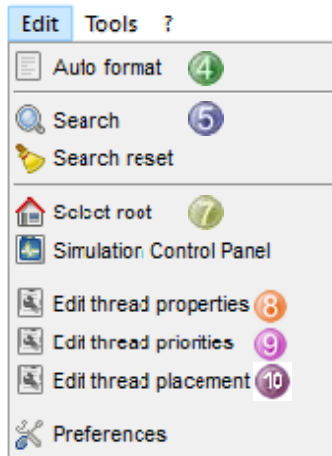
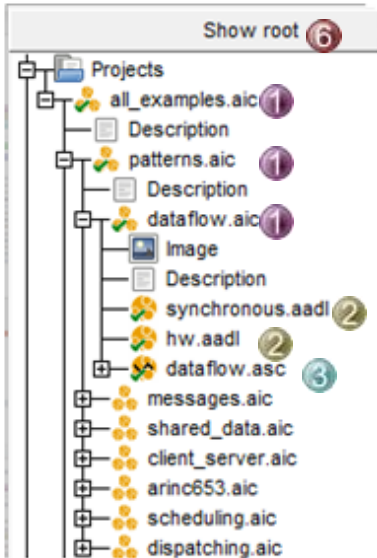
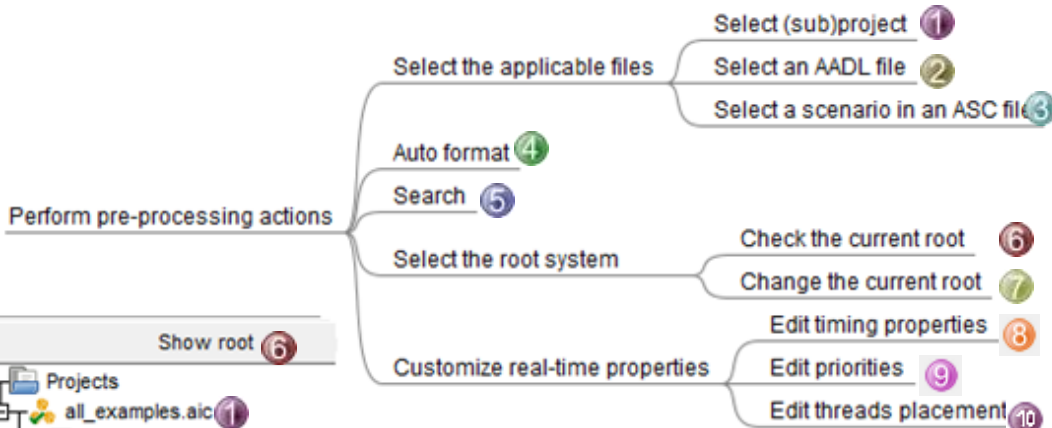
These examples can be directly loaded into AADL Inspector:

- patterns.aic:
Contains seven sub-projects listed below.

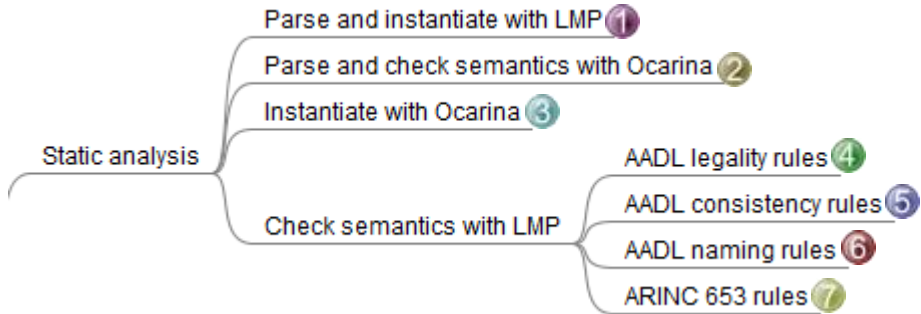
3. View environment



4. Pre-Processing



5. Static Analysis



Tools	?
Static Analysis	Parse and Instantiate (LMP) 1
LAMP Lab	Parse (Ocarina) 2
Timing Analysis	Instantiate (Ocarina) 3
Safety & Security Analysis	Check Consistency Rules (LMP) 4
Code Generation	Check Legality Rules (LMP) 5
	Check Naming Rules (LMP) 6
	Check ARINC 653 Rules (LMP) 7

5. Static Analysis example

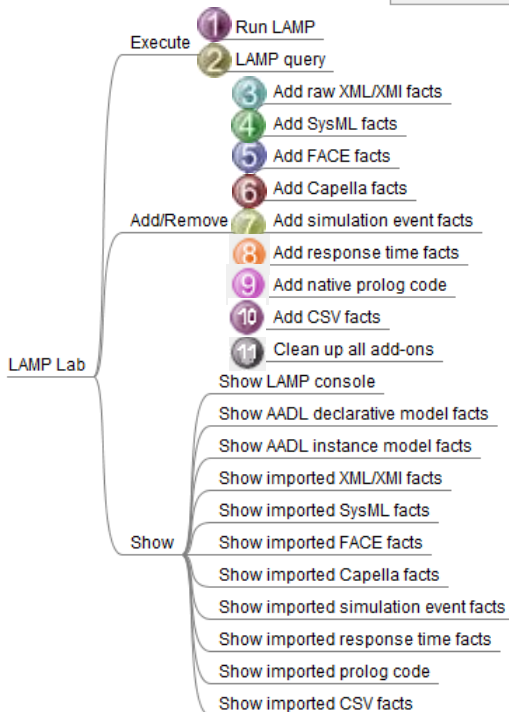
```
Static Analysis | LAMP Lab | Timing Analysis | Safety & Security Analysis | Cod
ME OP OI CC LC MC 653
-----
aadlrev2.17 (c)Ellidiss Technologies 06Apr2023
AADL-2.3 + BA-2.0
-----
the reference time unit is: ms
-----
*** INSTANCE MODEL ***
-----
Root System Instance: dataflow_Diag::dataflow.diagram
-----
110 (system)..... root
157 (processor)... root.my_platform.cpu (RM)
 20 (process)..... root.my_process
 35 (thread)..... t1 (PERIODIC)
 84 (subprogram)..... _square(28637)
 41 (thread)..... t2 (PERIODIC)
 84 (subprogram)..... _square(28637)
 47 (thread)..... t3 (PERIODIC)
 84 (subprogram)..... _square(28637)
 54 (thread)..... t4 (PERIODIC)
 84 (subprogram)..... _square(28637)
-----
```

6. LAMP Lab

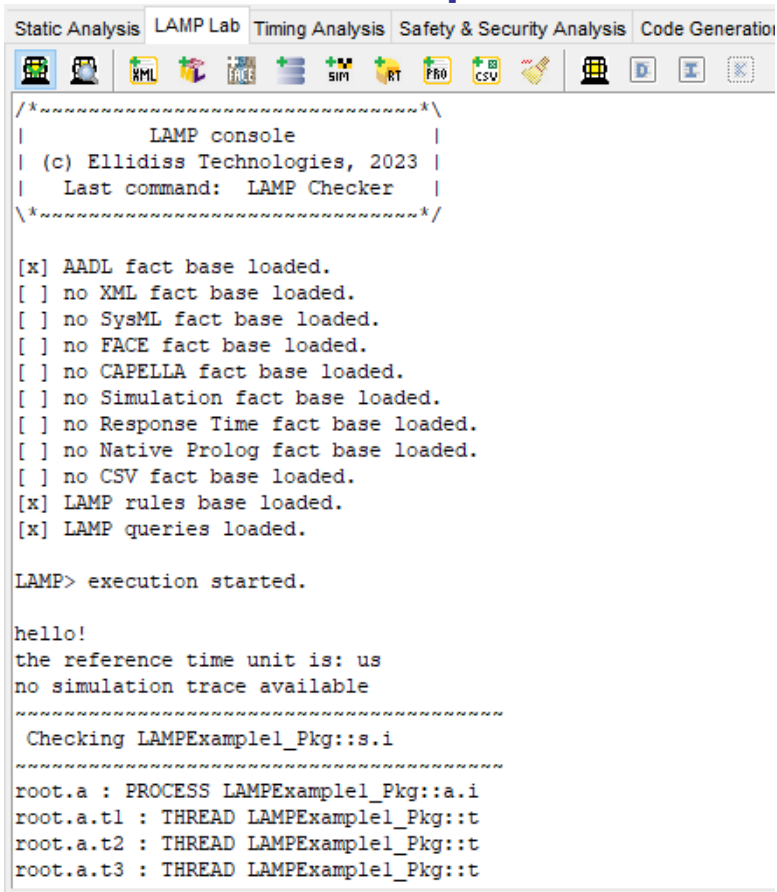


Tools ?

- Static Analysis
- LAMP Lab**
 - Run LAMP 1
 - LAMP query 2
 - Add raw XML/XMLI facts 3
 - Add SysML facts 4
 - Add FACE facts 5
 - Add CAPELLA facts 6
 - Add simulation events facts 7
 - Add response time facts 8
 - Add native prolog code 9
 - Add CSV facts 10
 - Clean up all add-ons 11
- Timing Analysis
- Safety & Security Analysis
- Code Generation



6. LAMP Lab example



The screenshot shows the LAMP Lab software interface. The title bar includes tabs for Static Analysis, LAMP Lab, Timing Analysis, Safety & Security Analysis, and Code Generation. The LAMP Lab tab is active, displaying a toolbar with icons for various analysis tools. The main window shows a console output with the following text:

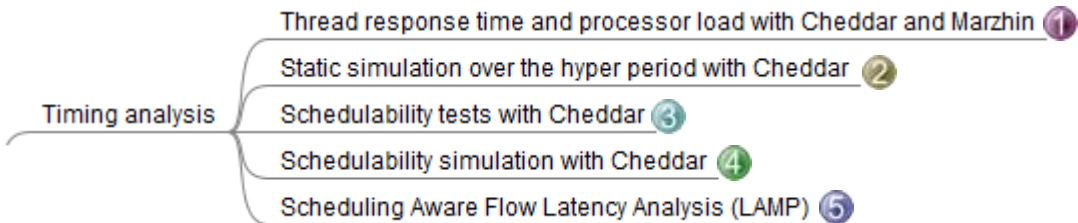
```
/*~*****~*/
|           LAMP console           |
| (c) Ellidiss Technologies, 2023 |
|   Last command: LAMP Checker   |
/*~*****~*/

[x] AADL fact base loaded.
[ ] no XML fact base loaded.
[ ] no SysML fact base loaded.
[ ] no FACE fact base loaded.
[ ] no CAPELLA fact base loaded.
[ ] no Simulation fact base loaded.
[ ] no Response Time fact base loaded.
[ ] no Native Prolog fact base loaded.
[ ] no CSV fact base loaded.
[x] LAMP rules base loaded.
[x] LAMP queries loaded.

LAMP> execution started.

hello!
the reference time unit is: us
no simulation trace available
*****
Checking LAMPExample1_Pkg::s.i
*****
root.a : PROCESS LAMPExample1_Pkg::a.i
root.a.t1 : THREAD LAMPExample1_Pkg::t
root.a.t2 : THREAD LAMPExample1_Pkg::t
root.a.t3 : THREAD LAMPExample1_Pkg::t
```

7. Timing Analysis



Tools ?


Static Analysis ▶


LAMP Lab ▶


Timing Analysis ▶


Safety & Security Analysis ▶


Code Generation ▶

 Processor Load & Thread Response Time Analysis ①

 Simulation Timelines (Cheddar) ②

 THE Theoretical Tests (Cheddar) ③

 SIM Simulation Tests (Cheddar) ④

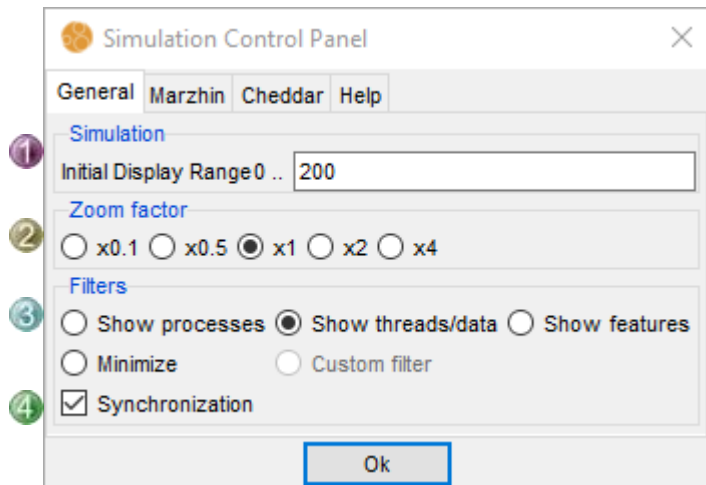
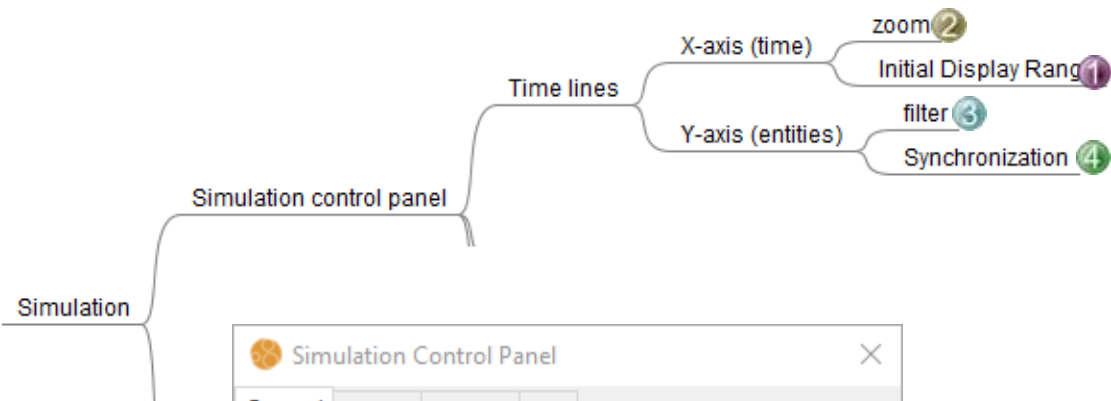
 Scheduling Aware Flows Latency Analysis (SAFLA) with LAMP ⑤

<http://beru.univ-brest.fr/cheddar/>

7. Timing Analysis example

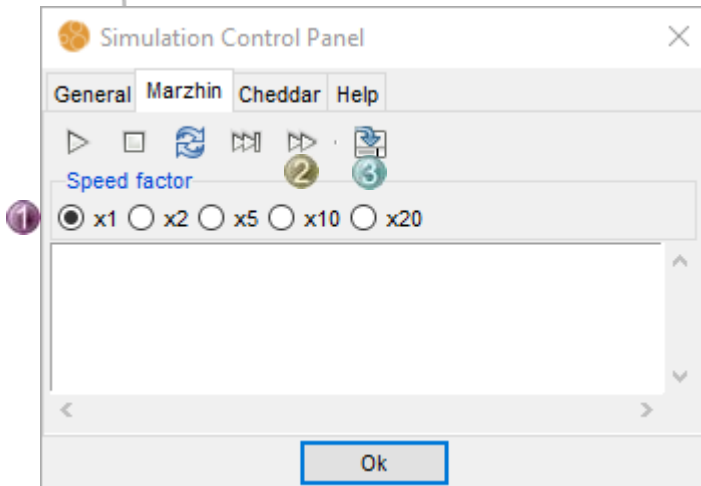
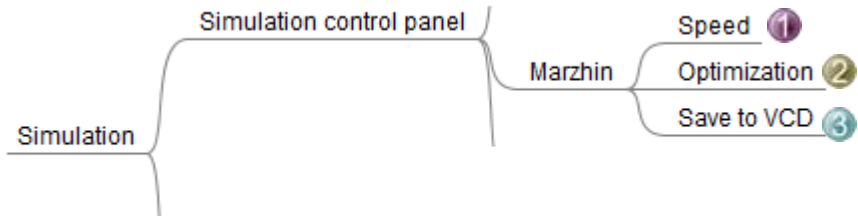
		Deadline	Computed	Max Cheddar	Max Marzhin	Avg Cheddar	Avg Marzhin	Min Cheddar	Min Marzhin
[-]	dashboard.dashboardcpu		30.00 %		31.40 %				
[-]	[-] dashboard.dashboardsw								
	[-] elaboratecommand	20	4.00000	4	4	4.00	4.00	4	4
	[-] displaystatus	10	2.00000	2	2	2.00	2.00	2	2
[-]	motors.motorscpu		56.67 %		58.96 %				
[-]	[-] motors.motorssw								
	[-] leftcontroller	15	7.00000	7	7	5.50	5.50	4	4
	[-] rightcontroller	15	5.00000	5	5	3.50	3.50	2	2
	[-] motorsmanager	10	3.00000	3	3	3.00	3.00	3	3
[-]	mainecu.maincpu		55.00 %		57.23 %				
[-]	[-] mainecu.mainsw								
	[-] controlmanager	20	8.00000	8	8	8.00	8.00	8	8
	[-] statusmanager	10	3.00000	3	3	3.00	3.00	3	3
[-]	can		80.00 %		79.77 %				
[-]	[-] VirtualLink								
	cnx_0		2.00000	11	9	11.00	6.22	11	4
	cnx_3		3.00000	9	10	9.00	4.65	9	3
	cnx_4		3.00000	6	10	6.00	5.29	6	3
	cnx_6		2.00000	13	5	13.00	3.11	13	2

8. Simulation timelines preferences



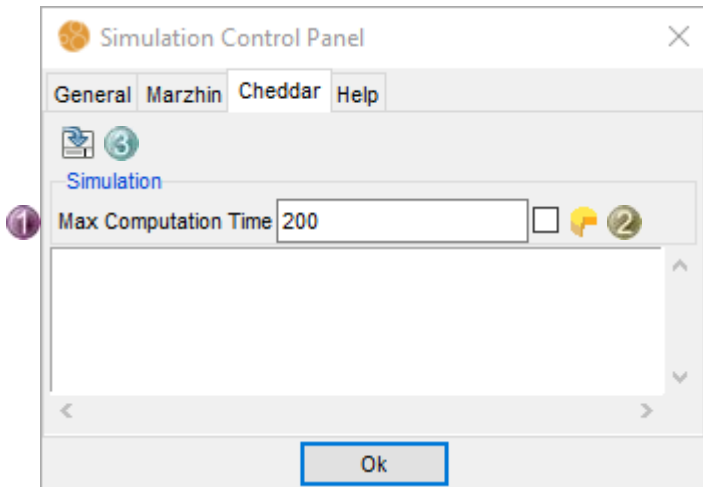
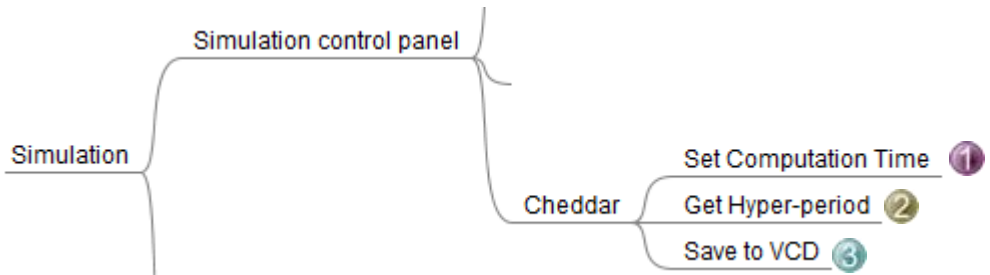
8. Simulation

Marzhin preferences

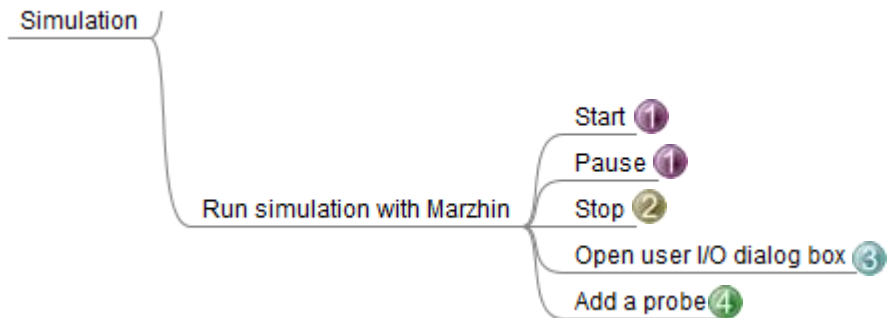


8. Simulation

Cheddar preferences



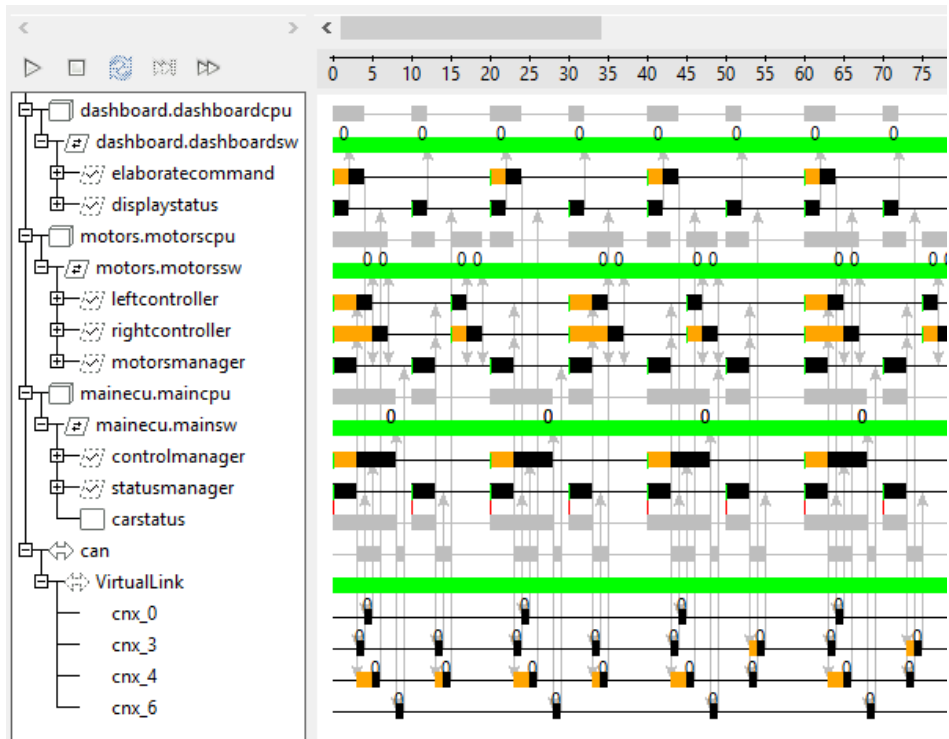
8. Simulation dashboard



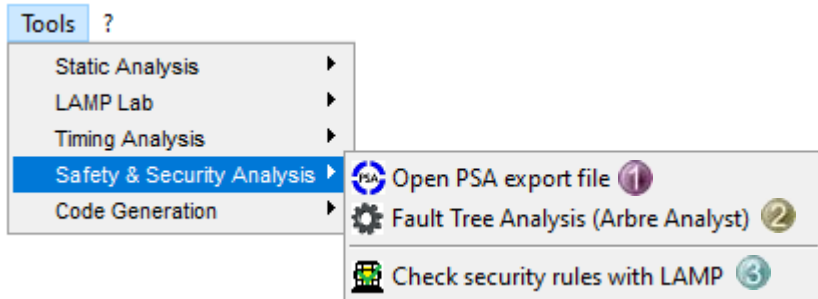
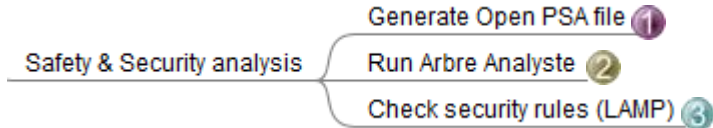
The screenshot illustrates the simulation dashboard interface. On the left, a component tree shows the hierarchy: `my_platform.cpu` (containing `my_process`), which in turn contains `input`, `output`, and four probes labeled `t1`, `t2`, `t3`, and `t4`. On the right, three windows are displayed:

- The `root.my_platform.cpu.my_process in/out ports` window shows the `input` data port set to 5 and the `output` data port set to 256, with `Close` and `Send all` buttons.
- The `output` window displays the current value of the output port, which is 256.
- The `t1 in my_process on my...` window displays a gauge for probe `t1`, with a needle indicating a value of approximately 10 on a scale from 0 to 20.

8. Simulation example

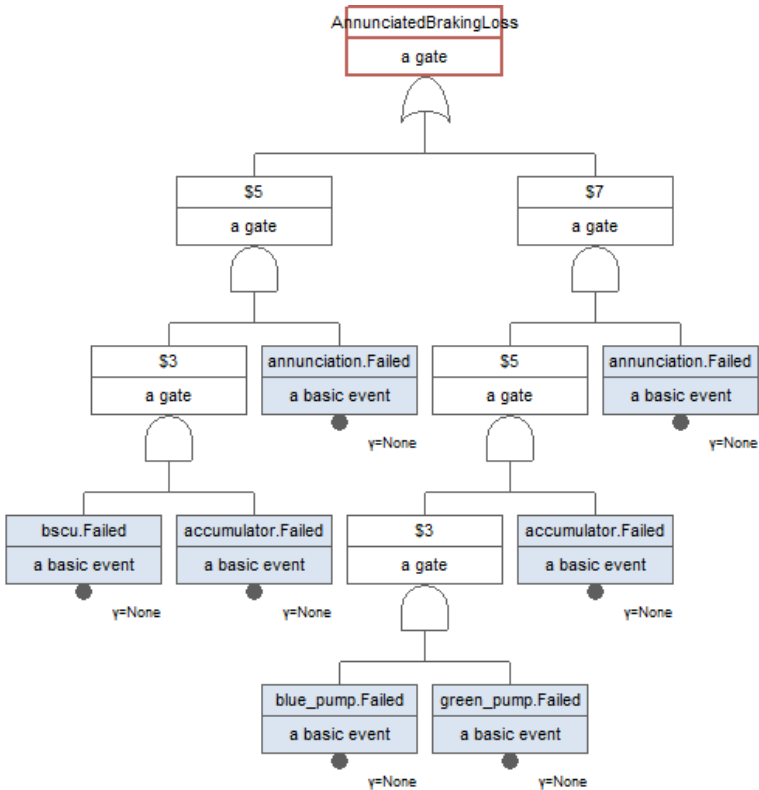


9. Safety & Security Analysis



9. Safety & Security Analysis

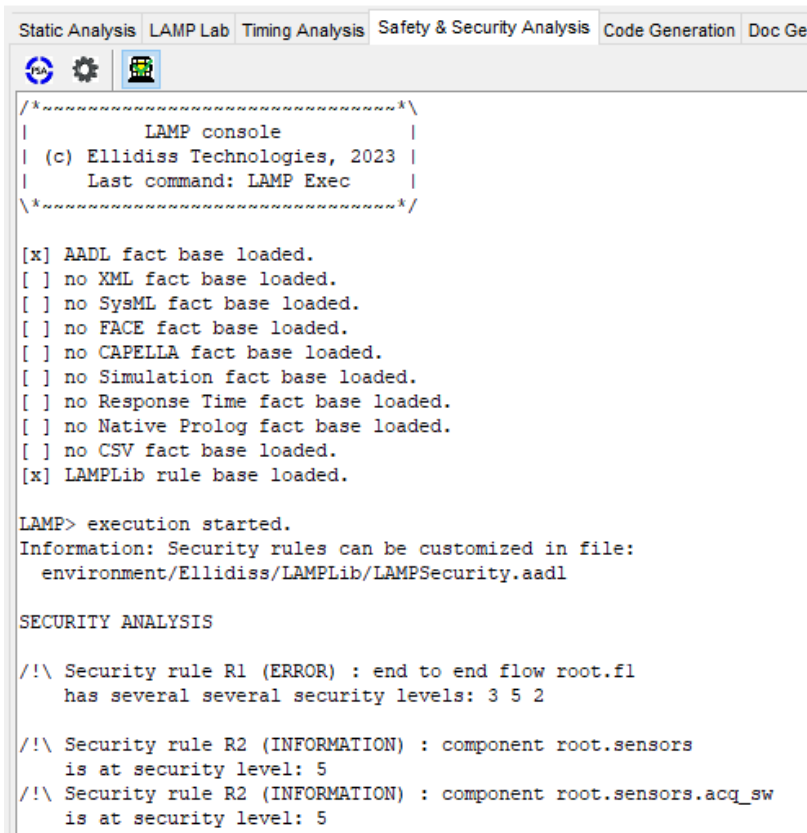
Fault Tree Analysis



<https://www.arbre-analyste.fr/en.html>

9. Safety & Security Analysis

Security Rules



```
Static Analysis | LAMP Lab | Timing Analysis | Safety & Security Analysis | Code Generation | Doc Ge
[SA] [G] [LAMP]
/*~~~~~*/
|           LAMP console           |
| (c) Ellidiss Technologies, 2023 |
|           Last command: LAMP Exec |
/*~~~~~*/

[x] AADL fact base loaded.
[ ] no XML fact base loaded.
[ ] no SysML fact base loaded.
[ ] no FACE fact base loaded.
[ ] no CAPELLA fact base loaded.
[ ] no Simulation fact base loaded.
[ ] no Response Time fact base loaded.
[ ] no Native Prolog fact base loaded.
[ ] no CSV fact base loaded.
[x] LAMPLib rule base loaded.

LAMP> execution started.
Information: Security rules can be customized in file:
  environment/Ellidiss/LAMPLib/LAMPSecurity.aadl

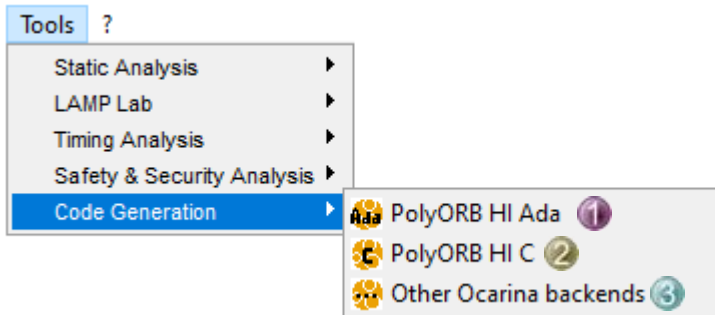
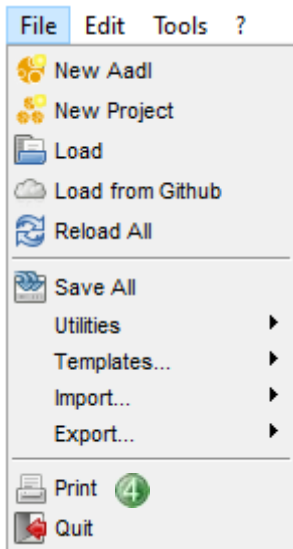
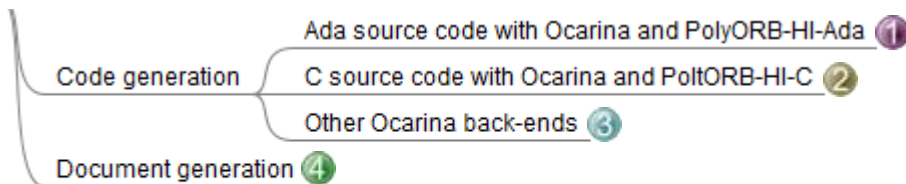
SECURITY ANALYSIS

/!\ Security rule R1 (ERROR) : end to end flow root.fl
  has several several security levels: 3 5 2

/!\ Security rule R2 (INFORMATION) : component root.sensors
  is at security level: 5

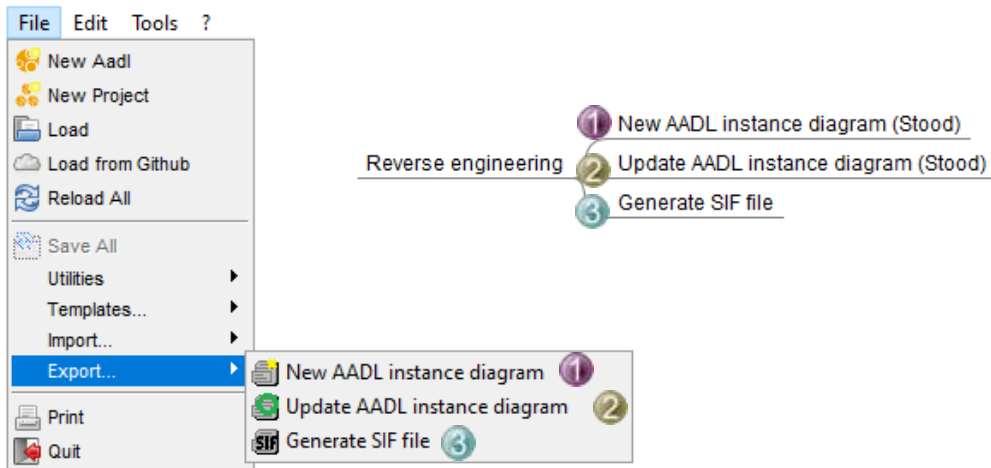
/!\ Security rule R2 (INFORMATION) : component root.sensors.acq_sw
  is at security level: 5
```

10. Code & Document Generation



<http://www.openaadl.org/ocarina.html>

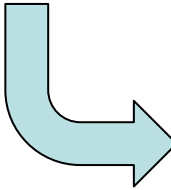
11. Reverse Engineering



11. Reverse Engineering example

The screenshot shows the AADL Inspector interface. The left pane displays a project tree with 'marx_pathfinder.aad' selected. The main pane shows the AADL code for 'sys_marx_pathfinder' and 'sys_marx_pathfinder_impl'. The right pane shows a 'Static Analysis' window with a timeline graph and a list of components and their states.

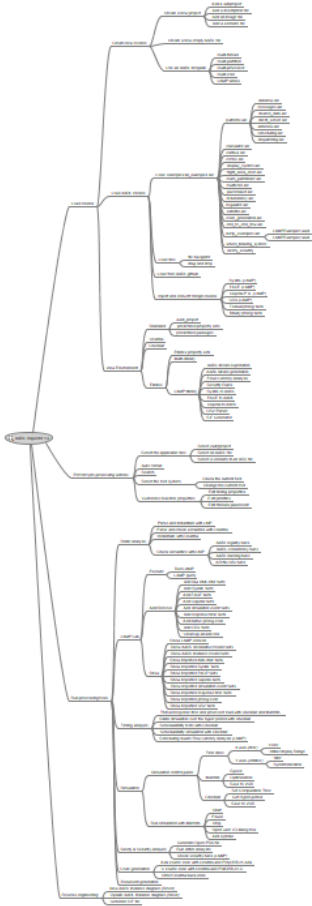
AADL Inspector



Stood for AADL

The screenshot shows a detailed component diagram for 'sys_marx_pathfinder'. The diagram illustrates the relationships between various components, including 'rs_6000', 'rs_1553', 'camera', 'accelerometer', 'metereological', 'star_analyser', 'thrusters', 'vavles', 'pathfinder_software', and 'pathfinder_hardware'. The diagram shows data and control flows between these components.

More Information



?	
Help	⌵ ? AADL Inspector 1.9 Quick Start
About	? AI User Manual
License info	? Cheddar
	? Consistency Rules
	? Legality Rules
	? Marzhin
	? Metrics
	? Naming Rules
	? ocarina
	? polyorb-hi-ada_ug
	? polyorb-hi-c_ug

<http://www.ellidiss.com>

aadl@ellidiss.com