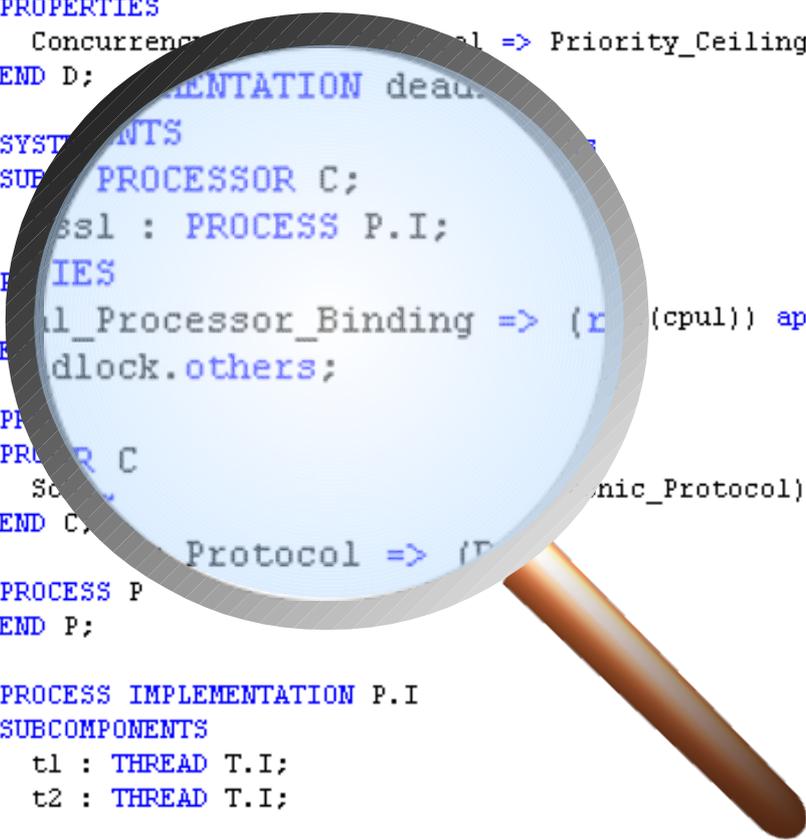




# AADL Inspector

Model analysis tool for the  
Architecture Analysis and Design Language

```
5 SYSTEM deadlock
6 END deadlock;
7
8 DATA D
9 -- deadlock occurs if concurrency control protocol is removed
10 PROPERTIES
11 Concurrency_Control => Priority_Ceiling_Protocol;
12 END D;
13 IMPLEMENTATION deadlock
14 SUBCOMPONENTS
15 SUBCOMPONENT PROCESSOR C;
16 Processor : PROCESS P.I;
17 PROPERTIES
18 Processor_Binding => (r (cpul)) applies to proces
19 deadlock.others;
20
21
22 PROCESSOR C
23 Processor : PROCESS P.I;
24 Processor_Binding => (r (cpul)) applies to proces
25 END C;
26 Priority_Ceiling_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;
```



- Multi-threaded software (RTOS)
- Time and Space Partitioned systems (TSP)
- Multi-processor distributed networks
- Multi-core architectures

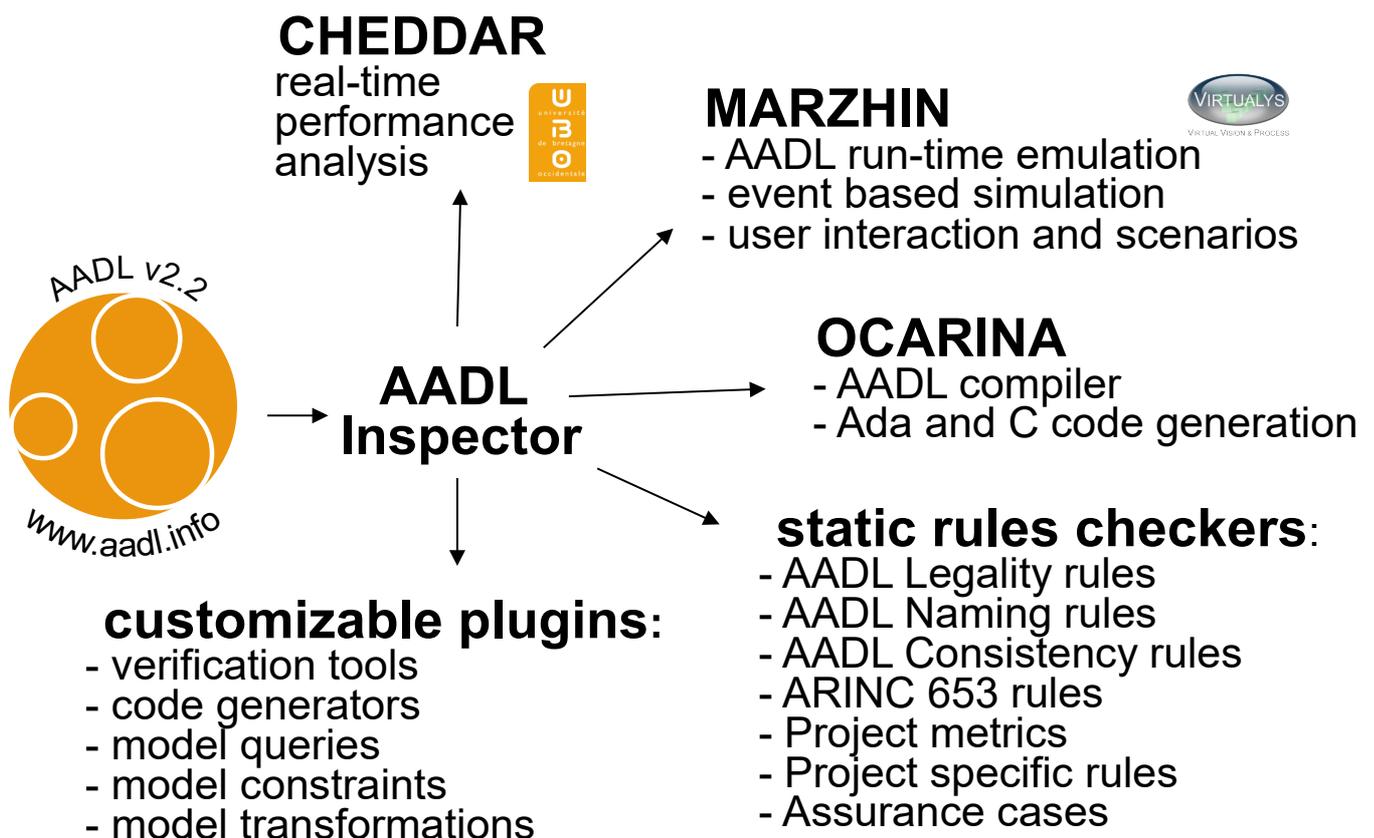
## AADL: designing scalable performance-critical real-time systems

Modeling and early validation of real-time features is a key concern for embedded applications. The Architecture Analysis and Design Language (AADL) is a textual and graphical language that can be used to design and analyse the software and hardware architecture of performance-critical real-time systems. AADL v2.2 is the last published version by the SAE, avionics systems division, under number AS 5506C. AADL Inspector can load the multiple textual AAD files or projects that contribute to a system specification and can also analyse AADL Behavior Annex and Error Annex extensions.

Thanks to its textual syntax, AADL can be used as a first class front end modelling language for designing large scale software intensive systems or as a hidden pivot format in case of use of another model driven modelling approach (UML, SysML, Capella, EEA, DSL, ...)

### Early detection of problems

One of the benefits of using the AADL for modeling critical systems is that this language has been designed in the purpose of being verified. AADL inspector has been specifically developed to insure that the model is correctly formed according to the AADL semantics, allowing an early detection of problems in the design process. Analysis reports are provided by the various analysis tools that are encompassed by AADL Inspector and have been developed by Ellidiss and his partners.



### It doesn't include your model processing features yet ?

Thanks to the Logic Model Processing (LMP) technology that provides a powerful and formal way to connect remote tools, AADL Inspector can easily incorporate additional analysis or production plugins or customize the existing ones. All the appropriate support will be provided by our technical team to help you to introduce project specific rules checkers, to add a new AADL bridge to your favorite verification tool, or plug a target source code generator.

# Static Analysis

AADL Inspector complies with the most recent version of the AADL standard, including its annexes (Data Modeling Annex, Behavior Annex, ARINC 653 Annex and Error Annex). For upwards compatibility, older AADL files are also accepted. Syntactic analysis, static semantics verification and model instantiation are implemented by two technologies: **Ocarina**<sup>1</sup> and **LMP**.

The screenshot displays the AADL Inspector interface. On the left, a project tree shows the 'ecosolar.aadl' file. The main window shows the AADL code for 'PROCESS MotorsSW'. The right pane is divided into two sections: a table for static analysis results and a Gantt chart for timing analysis.

Component	Deadline	Computed	Max Cheddar	Max Marzhin	Avg Cheddar
dashboard.dashboardcpu		30.00 %		31.52 %	
dashboard.dashboardsv					
elaboratecommand	20	4.00000	4	4	4.00
displaystatus	10	2.00000	2	2	2.00
motors.motorscpu		56.67 %		58.29 %	
motors.motorssw					
leftcontroller	15	7.00000	7	5	5.50
rightcontroller	15	5.00000	5	7	3.50
motorsmanager	10	3.00000	3	3	3.00
mainecu.maincpu		55.00 %		56.01 %	

The Gantt chart below the table shows the execution timeline for various components over time (120 to 195). Components like 'displaystatus', 'motors.motorscpu', 'mainecu.mainsv', and 'VirtualLink' are highlighted with green bars, indicating their execution periods.

## Timing Analysis

AADL Inspector also includes two Real-Time analysis tools and their respective AADL LMP bridger:

- **CHEDDAR**<sup>2</sup> for schedulability analysis (feasibility tests and static simulation over the hyper-period). It supports RM, DM, HPF, EDF and LLF scheduling of periodic tasks.
- **MARZHIN**<sup>3</sup> for event-driven dynamic simulation. It supports RM, DM, HPF and EDF scheduling of periodic and non-periodic tasks, with possible interaction with the external environment (pre-defined scenarios or user interaction).
- Response Time analysis and CPU Load estimate.

## Custom Assurance Cases Tools



AADL Inspector provides the **LAMP** (Logic AADL Model Processing) engine to implement inline verification tools that are embedded within the AADL model as annexes. It uses the industry proven LMP technology. LMP model checkers have been qualified by Airbus for the DO 178 certification of A380 and A350 embedded software.

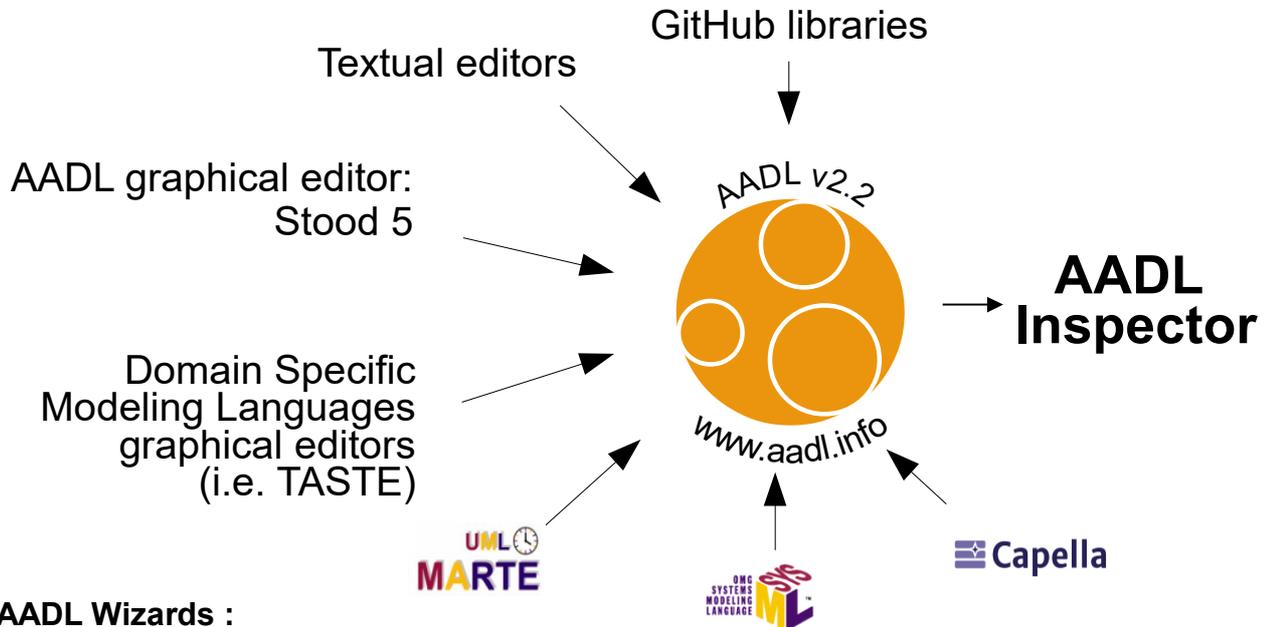
<sup>1</sup> Ocarina is developed by Telecom ParisTech, ISAE and ESA

<sup>2</sup> Cheddar is developed by the University of Brest

<sup>3</sup> Marzhin is developed by Ellidiss Technologies and Virtualys

## Inter-operability :

Thanks to the AADL textual notation, AADL Inspector can easily be integrated into heterogenous development tool-chains. Such textual specifications can be produced by AADL graphical editors such as Stood, Domain Specific editors or even a basic text editor. In addition, AADL source code can be produced by model transformations from other modeling languages such as SysML or UML/Marte. A direct access to AADL models libraries available on GitHub is also provided. The time-lines generated by the simulators can be saved in VCD (Value Change Dump) format for further processing.



## AADL Wizards :

AADL Inspector can also be used to dynamically modify your AADL project with data collected from the analysis tools, such as :

- Auto-format the AADL source text (pretty printer)
- Select the root system instance
- Set threads priorities according to the schedule policy
- Allocate threads onto available processors

## Platform Requirements :

AADL Inspector runs as a standalone executable on PC Windows or Linux.

A recent Java Run-time Environment (1.8 or more) is required to run the Marzhin simulator.

Less than 25 Mb free disk space is needed

French office:  
24 quai de la douane  
29 200 Brest  
France

+33 (0) 298 451 870



UK office:  
Mountbatten Court  
Worrall Street  
Congleton  
CW12 1DT  
UK  
+44 (0) 1260 291449