

# Ellidiss Technologies

[www.ellidiss.fr](http://www.ellidiss.fr)

Pierre Dissaux

AADL Demo Day

UAH, Huntsville, 14 Feb 2019



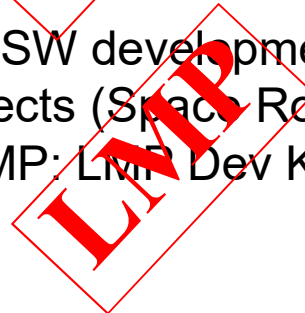
## 20 years tool support for major industrial projects:

- HOOD Software design tools for Ada and C
- Eurofighter Typhoon
- Airbus A340, A380, A400M, A350
- Tiger Helicopter (mission calculator)
- Rafale (engine control)

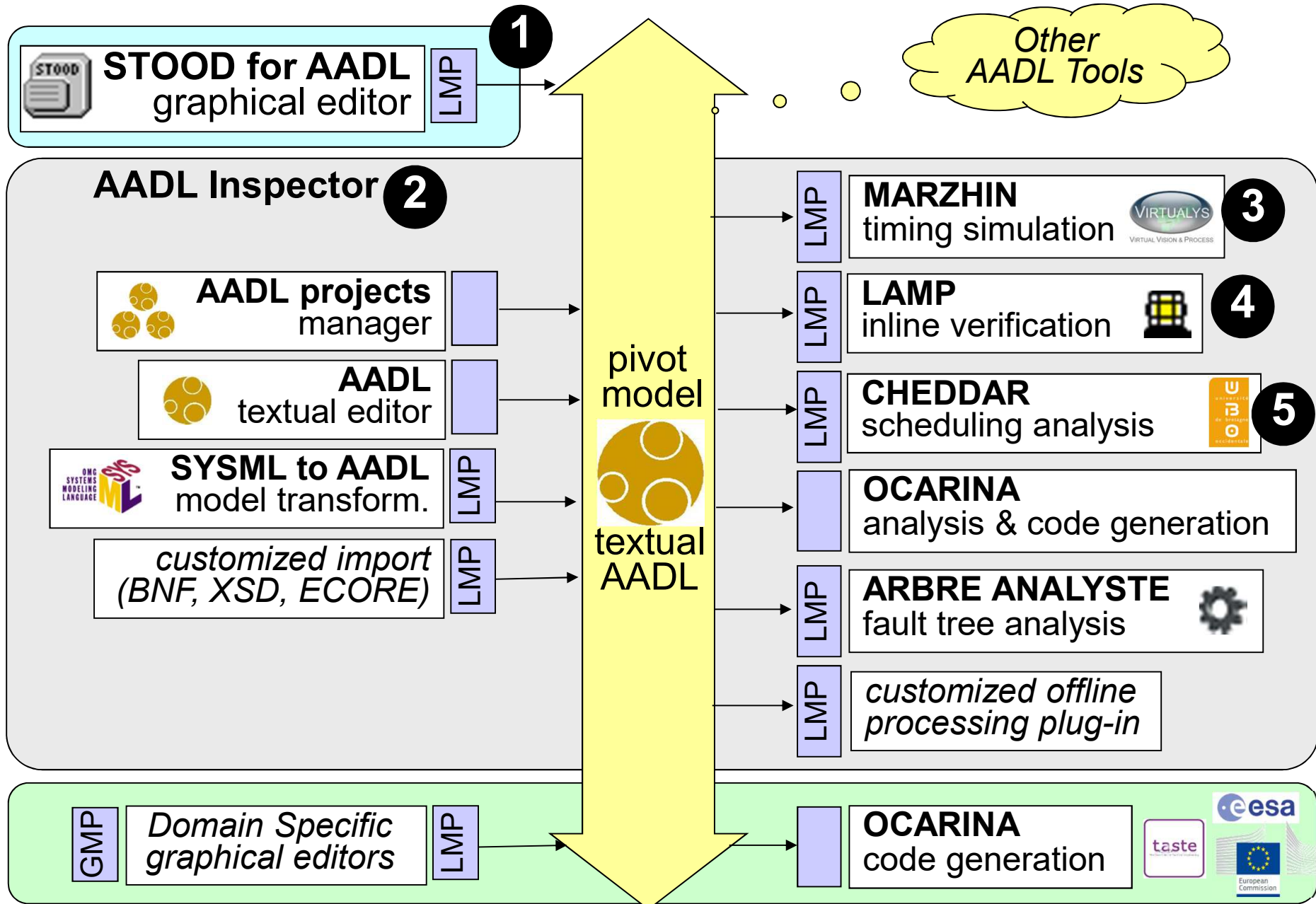


## 15 years investement in new tool technologies:

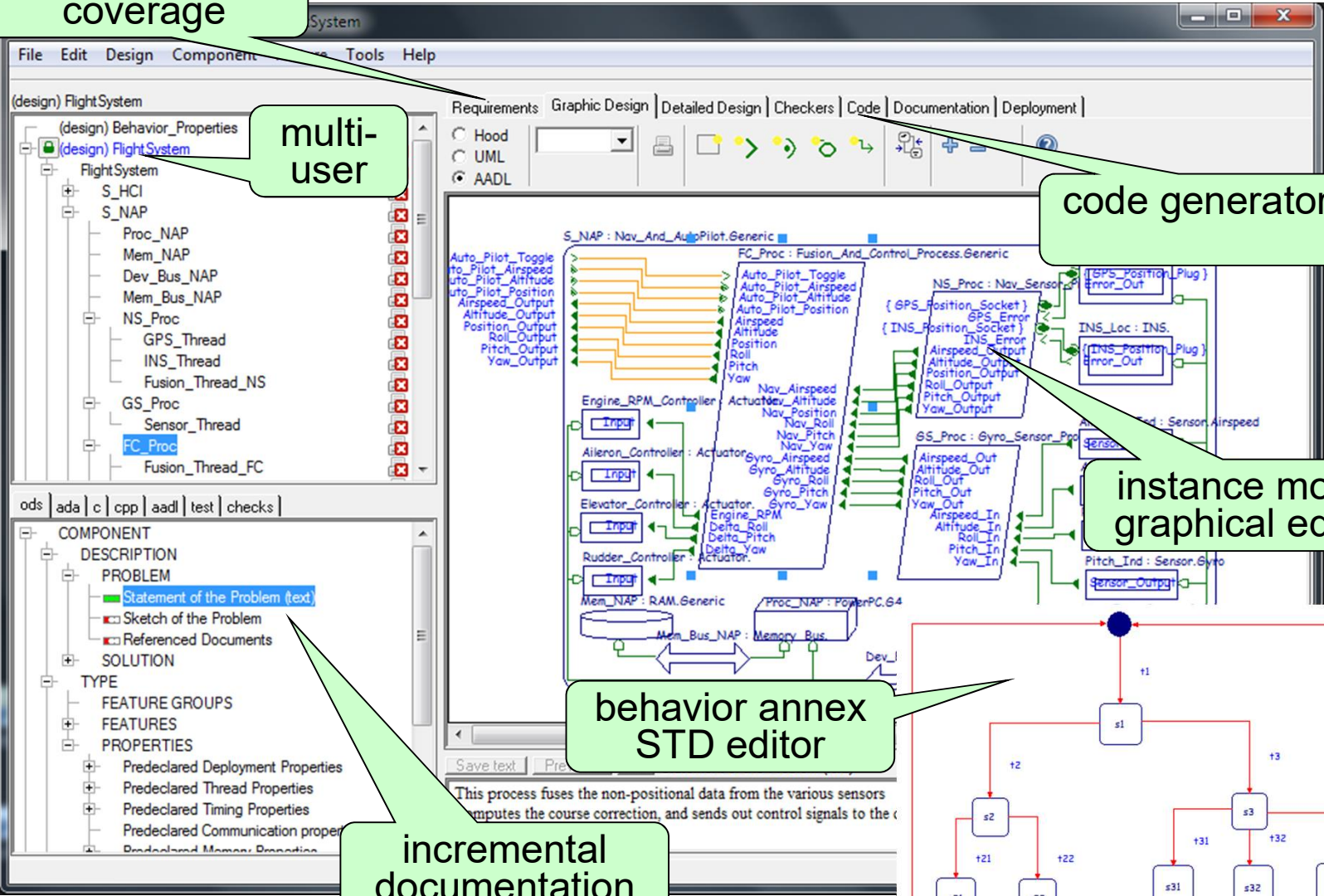
- SAE AS-5506: Architecture Analysis & Design Language
- AADL graphical modeling tools: Stood for AADL
- AADL analysis framework: AADL Inspector
- European Space Agency: TASTE Editors (Space SW development tools)
- European Commission: ERGO and MOSAR Projects (Space Robotics)
- Generic model processing technologies: GMP, LMP: LMP Dev Kit



# AADL centric tool-chains



# Stand for AADL key features



The screenshot shows the Stand for AADL software interface with several callouts highlighting key features:

- requirements coverage**: Callout pointing to the Requirements tab in the top menu.
- multi-user**: Callout pointing to the multi-user icon in the left sidebar.
- code generators**: Callout pointing to the Code tab in the top menu.
- instance model graphical editor**: Callout pointing to the main graphical diagram area.
- behavior annex STD editor**: Callout pointing to the bottom diagram area showing a state transition diagram.
- incremental documentation**: Callout pointing to the COMPONENT pane on the left.

The main graphical diagram shows a complex system architecture with components like `S_NAP`, `FC_Proc`, `NS_Proc`, `GS_Proc`, and various sensors and actuators. The bottom diagram is a state transition diagram with states `s1`, `s2`, `s3`, `s21`, `s22`, `s31`, `s32`, and `s33`, and transitions labeled with `t1` through `t330`.

Below the main diagram, there is a text box: "This process fuses the non-positional data from the various sensors and computes the course correction, and sends out control signals to the..."

# Top-Down modeling process for AADL

## Hierarchical Object Oriented Design (HOOD)

- Inherits 20 years usage by the biggest European avionics projects (Airbus, Eurofighter)
- Architectural Design (diagrams):
  - hierarchy of components with rigorous visibility rules
  - enable safe subcontracting (sub-trees)
  - ease testing, integration and maintenance
  - prevent from producing "spaghettiware"
- Detailed Design (structured text):
  - keep track of design decisions
  - requirements coverage
  - supporting framework for design documentation, coding and testing

## Benefits for the AADL user (Stood for AADL)

- Graphical editor of the AADL Instance Model (what you design is what you get)
- Data Hiding enforcement (visibility rules, no provides data access)
- AADL Declarative Model generator (textual AADL) for tools interchange
- Complement AADL design activities with detailed design (documentation and coding)

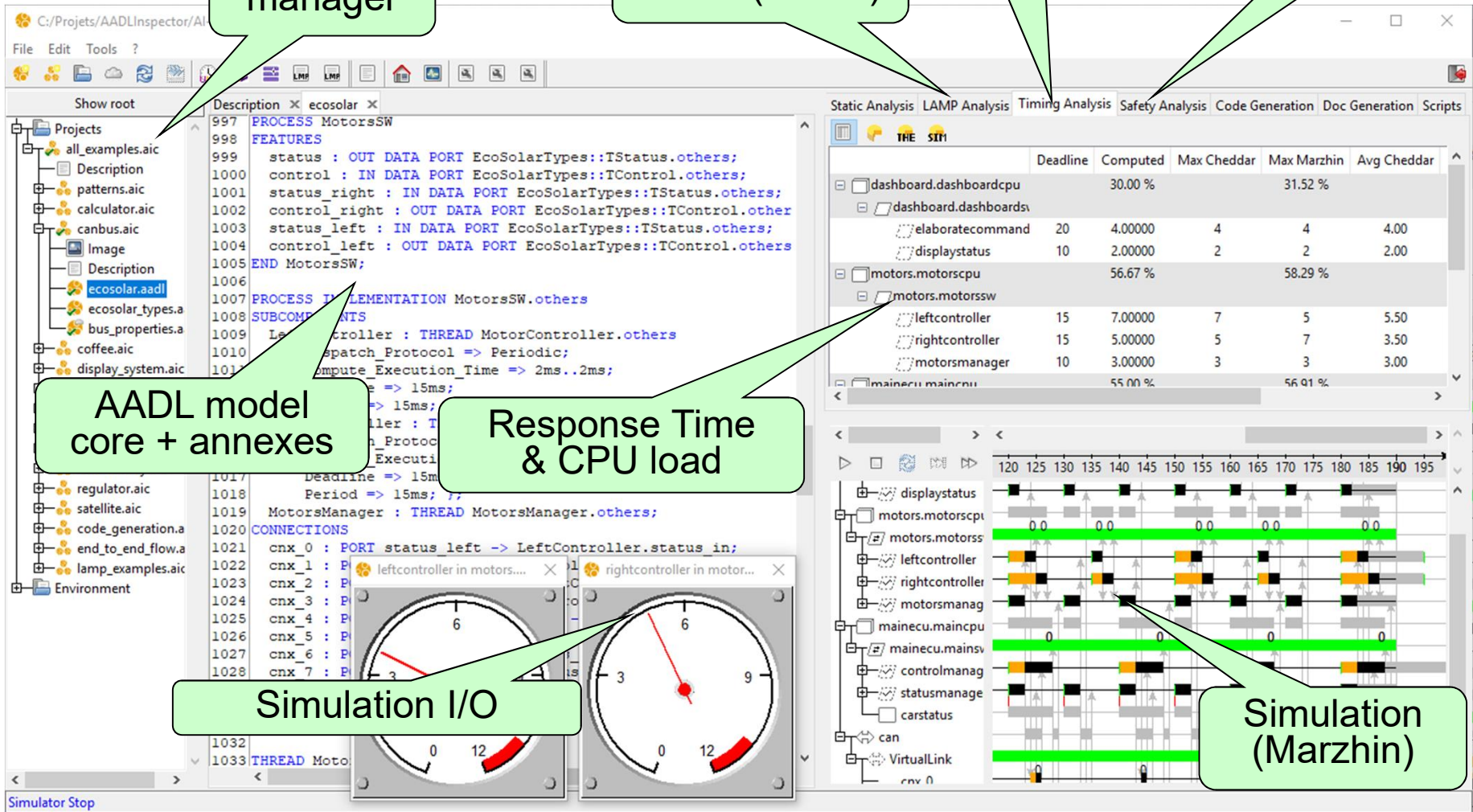
# AADL Inspector key features

Projects manager

Assurance cases (LAMP)

Timing analysis

Safety analysis



The screenshot displays the AADL Inspector interface with several key features highlighted by callouts:

- Projects manager:** A tree view on the left showing a project structure with folders like 'all\_examples.aic', 'patterns.aic', 'calculator.aic', 'canbus.aic', 'Image', 'ecosolar.aadl', 'ecosolar\_types.a', 'bus\_properties.a', 'coffee.aic', 'display\_system.aic', 'regulator.aic', 'satellite.aic', 'code\_generation.a', 'end\_to\_end\_flow.a', 'lamp\_examples.aic', and 'Environment'.
- AADL model core + annexes:** A central text editor showing AADL code for 'PROCESS MotorsSW' and 'IMPLEMENTATION MotorsSW.others', including features, subcomponents, and connections.
- Response Time & CPU load:** A table in the 'Timing Analysis' tab showing performance metrics for various components.
- Simulation I/O:** Two circular gauges at the bottom center representing simulation input/output values.
- Simulation (Marzhin):** A Gantt chart at the bottom right showing execution timelines for components like 'displaystatus', 'motors.motorscpu', 'motors.motorssw', 'leftcontroller', 'rightcontroller', 'motorsmanager', 'mainecu.maincpu', 'mainecu.mainsv', 'controlmanag', 'statusmanage', 'carstatus', 'can', and 'VirtualLink'.

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 %	

C:/Projets/AADLInspector/AI-1.7/examples/ecosolar/ecosolar.aadl

File Edit Tools ?

Show root

- Projects
  - all\_examples.aic
    - Description
    - patterns.aic
    - calculator.aic
    - canbus.aic
      - Image
      - Description
      - ecosolar.aadl
      - ecosolar\_types.a
      - bus\_properties.a
    - coffee.aic
    - display\_system.aic
    - flight\_deck\_door.a
    - mars\_pathfinder.ai
    - multicore.aic
    - pacemaker.aic
    - redundancy.aic
    - regulator.aic
    - satellite.aic
    - code\_generation.a
    - end\_to\_end\_flow.a
    - lamp\_examples.aic
  - Environment

Description x ecosolar x

```

997 PROCESS MotorsSW
998 FEATURES
999 status : OUT DATA PORT EcoSolarTypes::TStatus.others;
1000 control : IN DATA PORT EcoSolarTypes::TControl.others;
1001 status_right : IN DATA PORT EcoSolarTypes::TStatus.others;
1002 control_right : OUT DATA PORT EcoSolarTypes::TControl.others;
1003 status_left : IN DATA PORT EcoSolarTypes::TStatus.others;
1004 control_left : OUT DATA PORT EcoSolarTypes::TControl.others;
1005 END MotorsSW;
1006
1007 PROCESS IMPLEMENTATION MotorsSW.others
1008 SUBCOMPONENTS
1009 LeftController : THREAD MotorController.others
1010 { Dispatch_Protocol => Periodic;
1011   Compute_Execution_Time => 2ms..2ms;
1012   Deadline => 15ms;
1013   Period => 15ms; };
1014 RightController : THREAD MotorController.others
1015 { Dispatch_Protocol => Periodic;
1016   Compute_Execution_Time => 2ms..2ms;
1017   Deadline => 15ms;
1018   Period => 15ms; };
1019 MotorsManager : THREAD MotorsManager.others;
1020 CONNECTIONS
1021 cnx_0 : PORT status left -> LeftController.status in;
1022 cnx_1 : P leftcontroller in motors... x 1 rightcontroller in motor... x
1023 cnx_2 : P
1024 cnx_3 : P
1025 cnx_4 : P
1026 cnx_5 : P
1027 cnx_6 : P
1028 cnx_7 : P
1029 cnx_8 : P
1030 cnx_9 : P
1031 END MotorsS
1032
1033 THREAD Moto

```

Static Analysis LAMP Analysis Timing Analysis Safety Analysis Code Generation Doc Generation Scripts

THE SIM

	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 %	

Simulator Stop

# AADL Inspector 1.7

## AADL projects manager

- core 2.2 + annex sub-languages EMV1, EMV2, BA 2.0
- hierarchical AADL project structure:
  - AADL environment (libraries, property sets)
  - sharable sub-projects
  - simulation scenarios
  - documentation sections (text, pictures)

## Imports XML/XMI models

- generic transformation process for ECore based models using LMP
- existing prototypes for UML/MARTE, SysML, Capella, ...
- require precise mapping rules to be formalized (project dependent)

## AADL model processing

- turnkey embedded tools:
  - Cheddar (scheduling analysis)
  - Marzhin (event based simulation)
  - Ocarina (AADL compliancy analysis, code generation)
- user defined on-line assurance case checkers with the LAMP annex
- customizable off-line plug-ins using the LMP toolbox

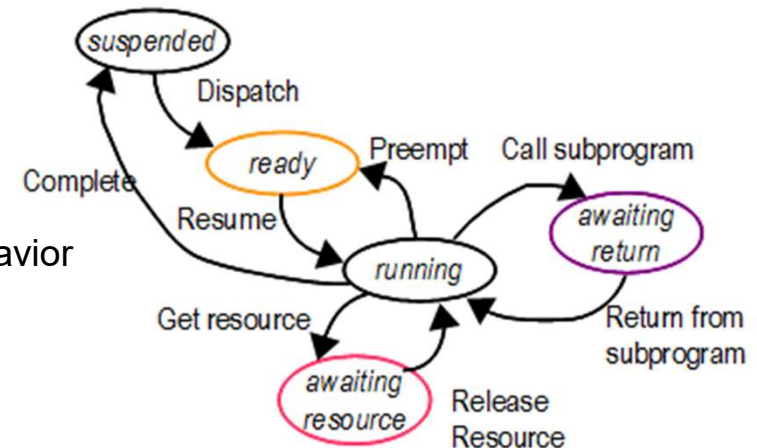




# Marzhin Executable AADL

## Multi-agent real-time simulator:

- Based on a pre-existing multi-agent kernel
- Specialized agents to represent real-time software constructs:
  - Processor and scheduler
  - Process and partition
  - Thread and shared data
  - Ports and connections
  - Bus and bus messages
- The agents interact together and exhibit a global behavior



## Implementation of the AADL run-time

- Standard AADL run-time semantic
- Behavior Annex interpreter
- Supports multi-processor, multi-partitions and multi-core architectures
- Generates system state changes events

## Accepts user interaction

- Can be controlled by scenarios or dialogs
- Used to display simulation traces
- Used to animate 2D/3D graphics



## Logical AADL Model Processing:

- Constraint language
- Assurance cases
- Inline verification rules
- Fully integrated within AADL models (LAMP Annex)
- Can replace REAL, LUTE, AGREE, RESOLUTE, ...by a single one

## LAMP features:

- Standard prolog language:
  - No new language to define & learn & maintain
  - Declarative syntax and formal semantics (ISO/IEC 13211)
  - Byte code available for IP libraries
- Exhaustive AADL model accessors:
  - Core language
  - Behavior Annex
  - Error Annex
- LAMPLib: predefined rules library
- Can process other input data sets (requirements, analysis results, ...)
- Available in AADL Inspector 1.7 (LAMP Checker)



```

THREAD t
FEATURES
  i : IN DATA PORT d;
  o : OUT DATA PORT d;
PROPERTIES
  DISPATCH PROTOCOL => Periodic;
  PERIOD => 15ms;
  DEADLINE => 8ms;
  COMPUTE_EXECUTION_TIME => 2ms..2ms;
  MY_PROPERTIES::MAX_VALUE => 80 APPLIES TO o;
ANNEX Behavior_Specification {**
  VARIABLES
    v : d;
  STATES
    s : INITIAL COMPLETE FINAL STATE;
  TRANSITIONS
    t : s -[ON DISPATCH]-> s { rand!(v); computation(10ms); o = v };
  
```

access to AADL property values

```

ANNEX LAMP {**
  checkOverflow(Id,Class) :-
    concat(Id, '.o', F),
    getProperties(F,Class, 'MY_PROPERTIES::MAX_VALUE', M),
    getPortValue(F, T, V),
    strToNum(V, W), strToNum(M, N), W > N,
    write('overflow of out data port '),
    write(F), sp, write(' at tick '), write(T), nl,
    write('( '), write(W), write(' > '), write(80), write(')'), nl,
    fail.
  checkOverflow(Id,Class).
**}
  
```

access to AADL simulation output

overflow detection

# Cheddar: Experimental Multiprocessor & Multicore Scheduling Analysis

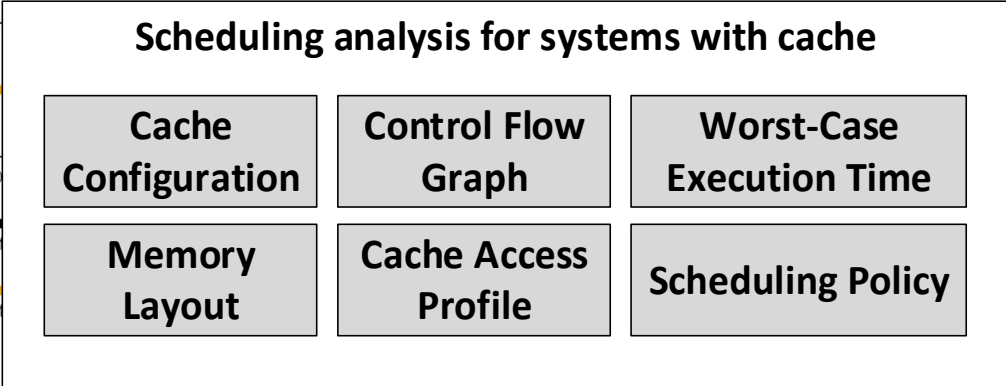
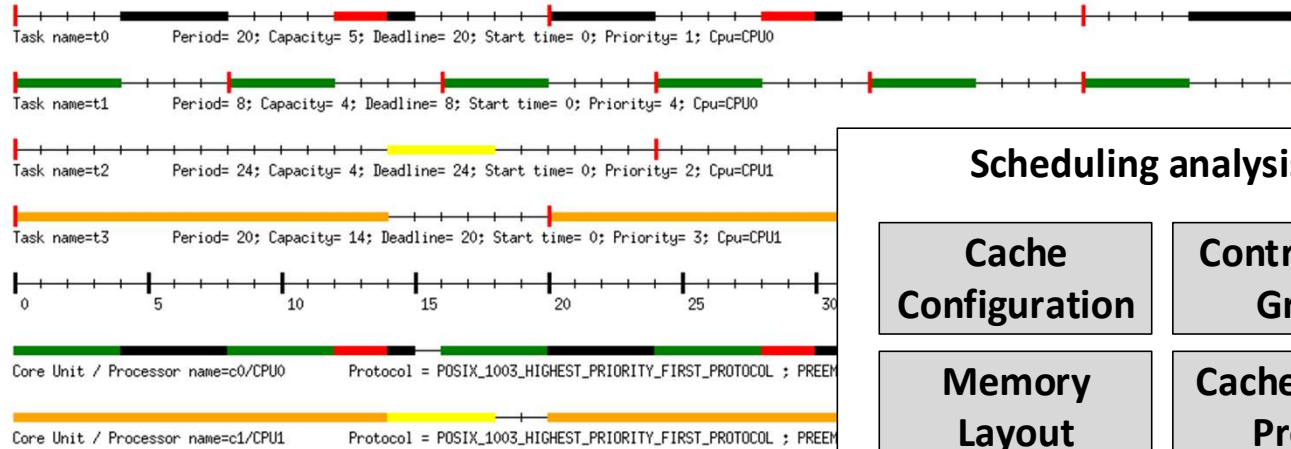
## Started during the SMART project (completed in 2014):

- How to model multicore/multiprocessor architectures with AADL
- Choose or design new scheduling analysis methods for such architectures

## Main multicore/multiprocessors features of Cheddar 3.x:

- **Partitioned and global scheduling policies:** extension of classical uniprocessor policies such as fixed priority or EDF + specific multicore policies such as Proportional Fair, EDZL, LLREF, ...
- **Design of PAES partitioning algorithms:** trade-off between preemption, latency, communication, ...
- **Support of shared resources between cores:** cache, network of chips

# Example: Cache-Aware Scheduling Analysis



## Scheduling simulation with cache:

- L1 uniprocessor instruction caches
- Sustainable CPRD model (Cache Preemption Related Delay)
- And known feasibility interval (proved):  $[0, LCM(P_i)]$

## Cache-Aware Priority Assignment Algorithm:

- Audsley oriented algorithm

# Conclusion

## AADL commercial tools

- Stood for AADL: instance model graphical editor for AADL
- AADL Inspector: analysis and simulation

## Technology

- LMP: model processing toolbox (prolog)
- GMP: DSL graphical editor framework
- Research collaboration with University of Brest/Lab-STICC

## Services

- Tools support
- AADL consulting
- Graphical front ends development
- Model processing tools (rules checkers, generators)
- Model transformations
- Heterogeneous tools integration
- R&D partnerships