

AADL Tools

AADL Committee, Chattanooga
May 5th, 2016

Pierre Dissaux





UK based company
aka. TNI Europe Ltd
Tools sales office



Fr based company
New tools development
R&D center

20 years + support to 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)



10 years + investement in new technology:

- SAE AS-5506: Architecture Analysis & Design Language
- AADL graphical modeling tools: Stood for AADL
- AADL analysis framework: AADL Inspector
- European Space Agency (TASTE Frame Contract)
- DSM graphical editors: TASTE, COMPASS,...
- Generic model processing technologies: GMP, LMP



Tools for AADL

- **Design: Stood for AADL**

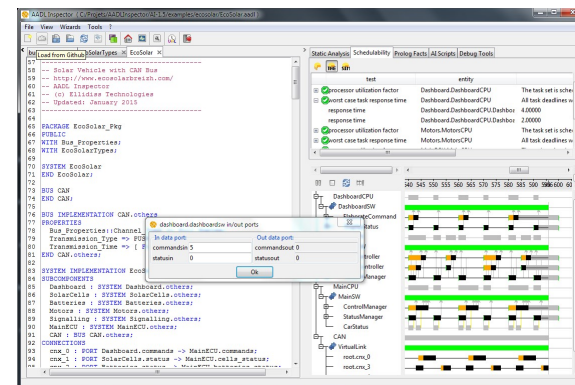
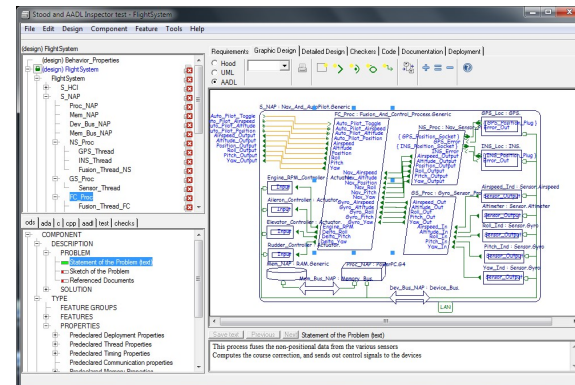
- AADL project management
- AADL Instance Model graphical editor
- Requirement traceability
- Documentation generator
- Export textual AADL

- **Verification: AADL Inspector**

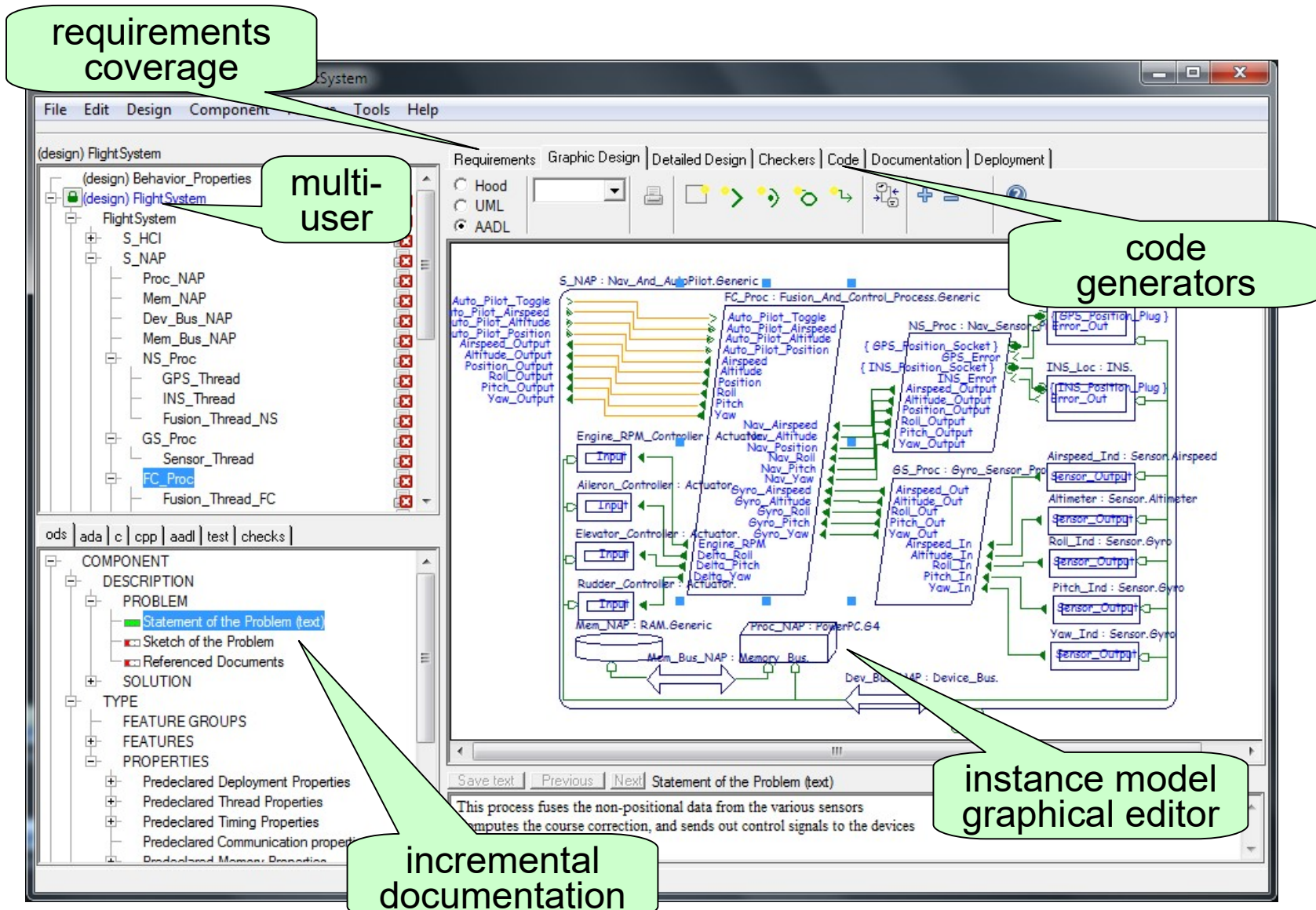
- Import textual AADL
- Model processing plugins
 - Static rules checkers
 - Scheduling analysis (Cheddar)
 - Simulation (Marzhin)
- Pre-processors:
 - Import UML profiles (MARTE, SysML, ...)
 - Import Domain Specific Models (XML)

- **Model Processing Toolbox: LMP**

- Supported languages: AADL, Ada, C, XM* (XML, XMI, ECore)
- Implementation: parsers + prolog engine and libraries



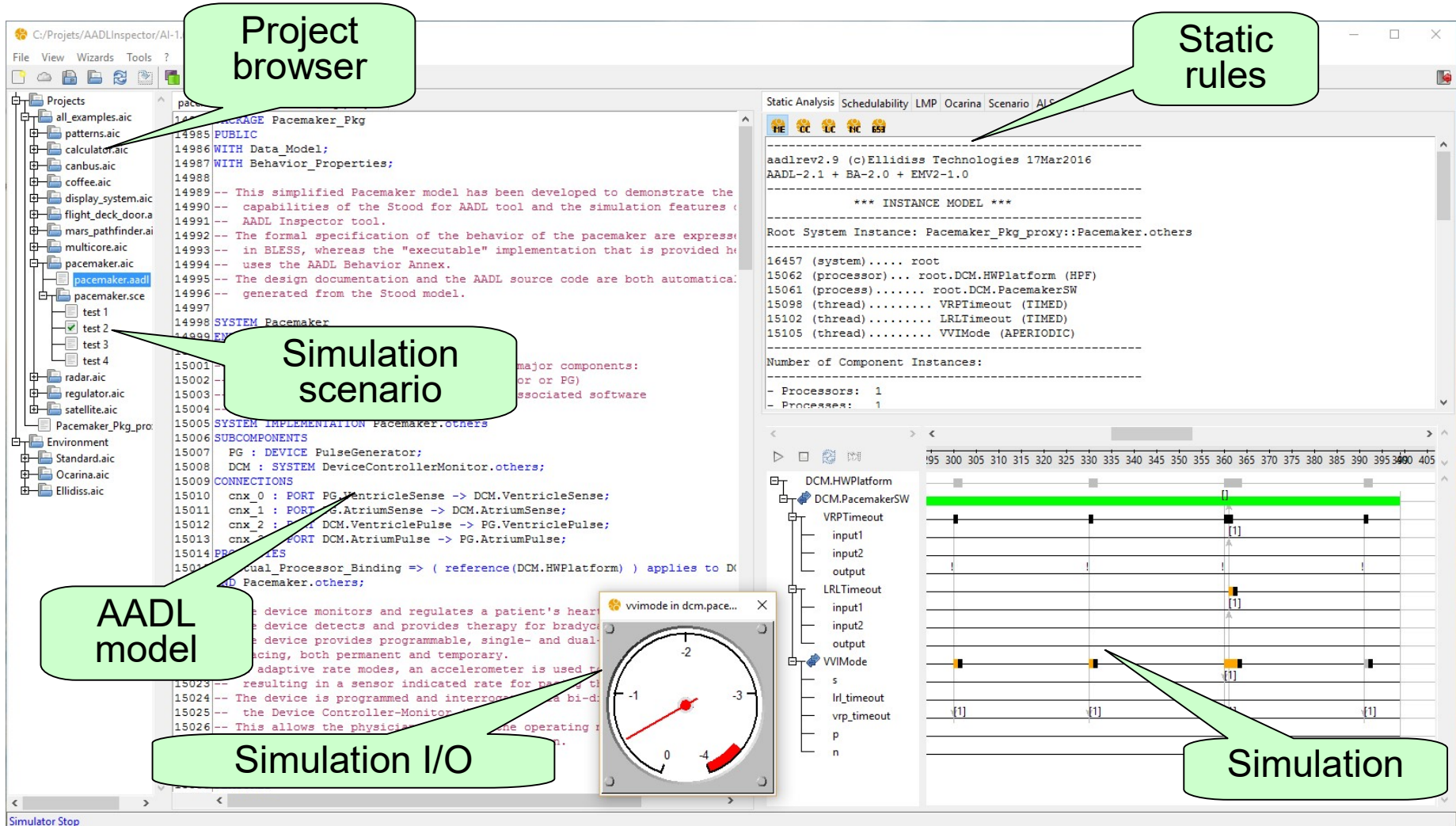
Stood for AADL



Top-Down modelling process for AADL

- Offers an industry proven practical modeling process to AADL designers
- Hierarchical Object Oriented Design (HOOD)
 - Inherits 20 years usage for 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
- Fully applicable to AADL (cf. Stood for AADL)
 - Graphical editor of the AADL Instance Model (what you design is what you get)
 - AADL Declarative Model generator (textual AADL)
 - Lower level software architecture can be refined with standard HOOD language to support the Detailed Design process and coding.

AADL Inspector



The screenshot displays the AADL Inspector application window, which is divided into several panes. The left pane shows a **Project browser** with a tree view of project files, including 'all_examples.aic', 'patterns.aic', 'calculator.aic', 'canbus.aic', 'coffee.aic', 'display_system.aic', 'flight_deck_door.aic', 'mars_pathfinder.aic', 'multicore.aic', 'pacemaker.aic', 'pacemaker.aadl', 'pacemaker.sce', 'test 1', 'test 2', 'test 3', 'test 4', 'radar.aic', 'regulator.aic', 'satellite.aic', 'Pacemaker_Pkg_proxy', 'Environment', 'Standard.aic', 'Ocarina.aic', and 'Ellidiss.aic'. The central pane displays the **AADL model** source code, which includes package declarations, system definitions, and component connections. The right pane shows the **Static rules** section, which includes a header for 'Static Analysis' and a list of rules such as 'aadrrev2.9 (c)Ellidiss Technologies 17Mar2016', 'AADL-2.1 + BA-2.0 + EMV2-1.0', and '*** INSTANCE MODEL ***'. Below this, the 'Root System Instance' is defined as 'Pacemaker_Pkg_proxy::Pacemaker.others'. The bottom right pane displays the **Simulation** results, which include a timeline view showing the execution of various components over time. A callout box labeled 'Simulation I/O' points to a small window titled 'vvmode in dcm.pace...' which shows a circular gauge with a needle pointing to approximately -2.5.

Project browser

Static rules

Simulation scenario

AADL model

Simulation I/O

Simulation

Model Processing Framework

- Imports AADL textual specifications
 - core 2.2 + annex sub-languages EMV1, EMV2, BA 2.0
 - interface with other AADL editors (Osate, Stood, ...) and github access
 - enhanced project management (hierarchical project structure, scenarii, ...)
- Imports XML/XMI models
 - generic transformation process into AADL 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)
 - customizable plugins using the LMP AADL toolbox:
 - AADL parser (aadlrev)
 - AADL processing libraries
 - prolog engine



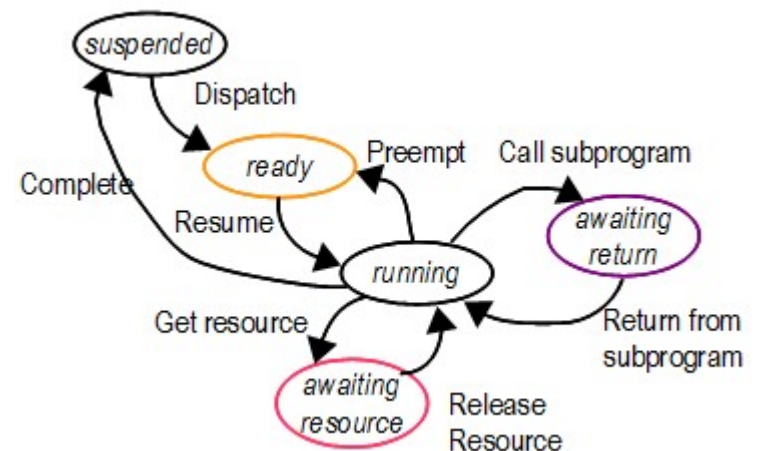
Marzhin

Executable AADL

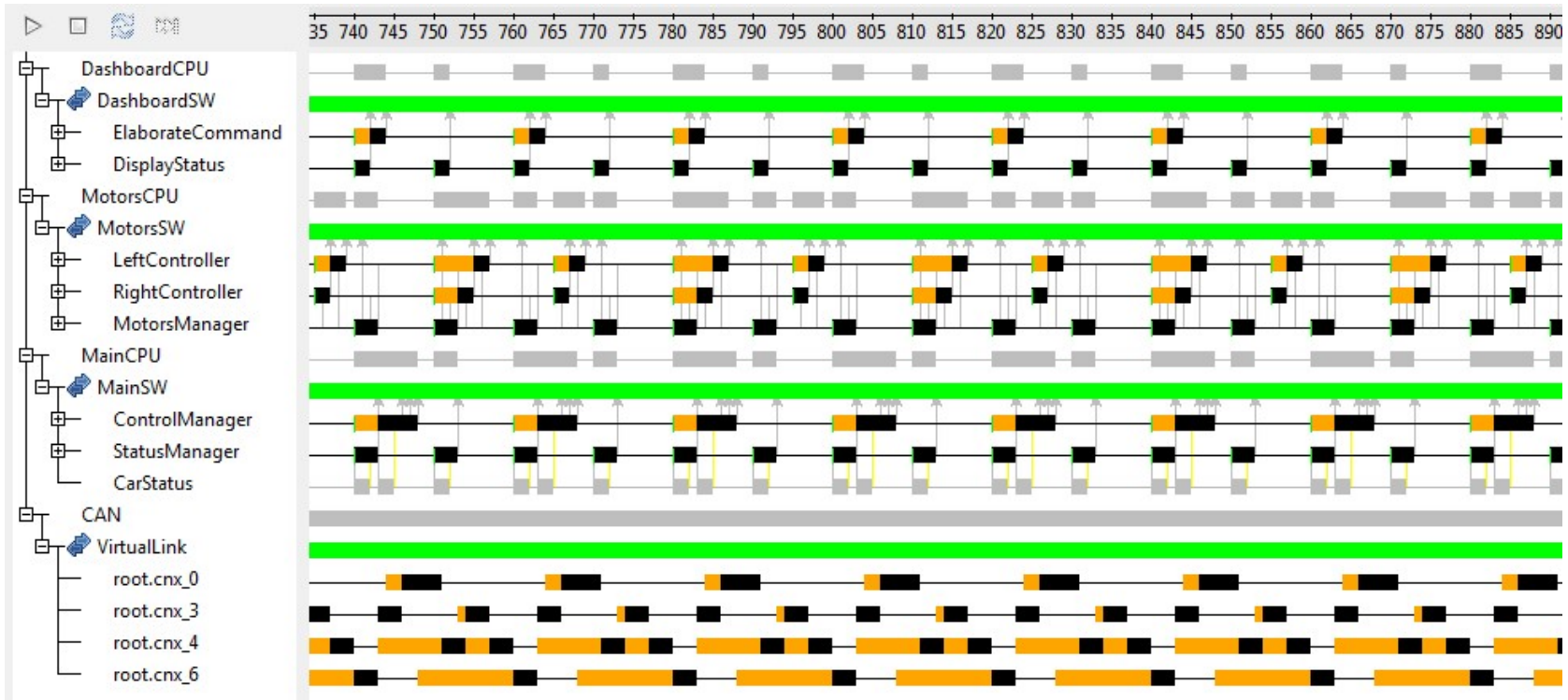
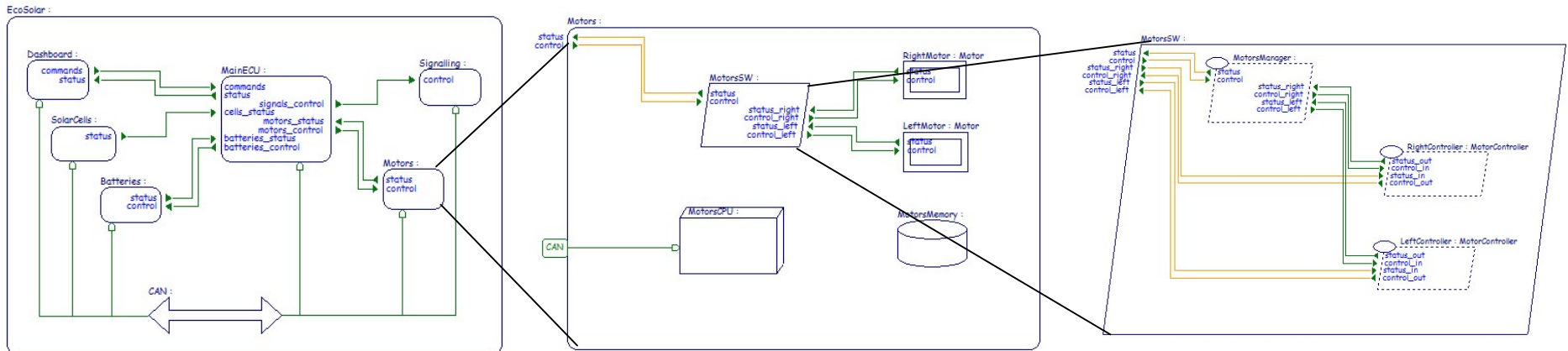
- Multi-agent real-time simulator:
 - Based on a pre-existing multi-agent kernel
 - Specialized agents to represent real-time software constructs:
 - Processor and scheduler
 - Process and partition
 - Thread and shared data
 - Ports and connections
 - Bus and bus messages
 - The agents interact together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

- Implementation of the AADL run-time
 - Standard run-time semantic
 - Details with the Behavior Annex
 - Example: an AADL thread:

- Generates system state changes events
 - Accepts user interaction
 - Can be controlled by scenari
 - Used to display simulation traces
 - Used to animate 2D/3D graphics

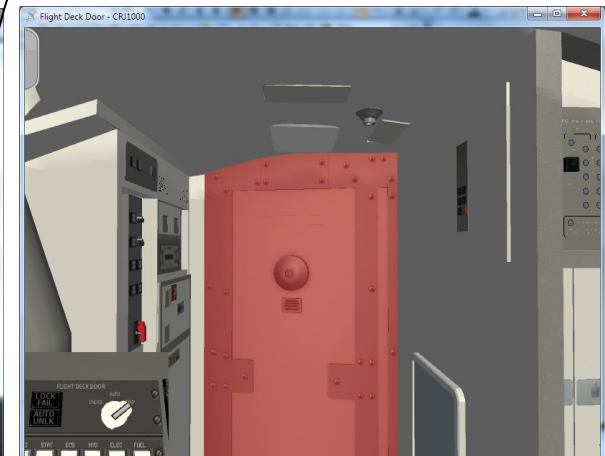
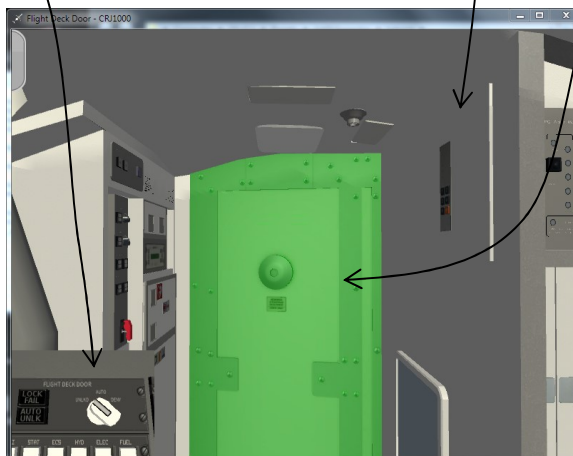
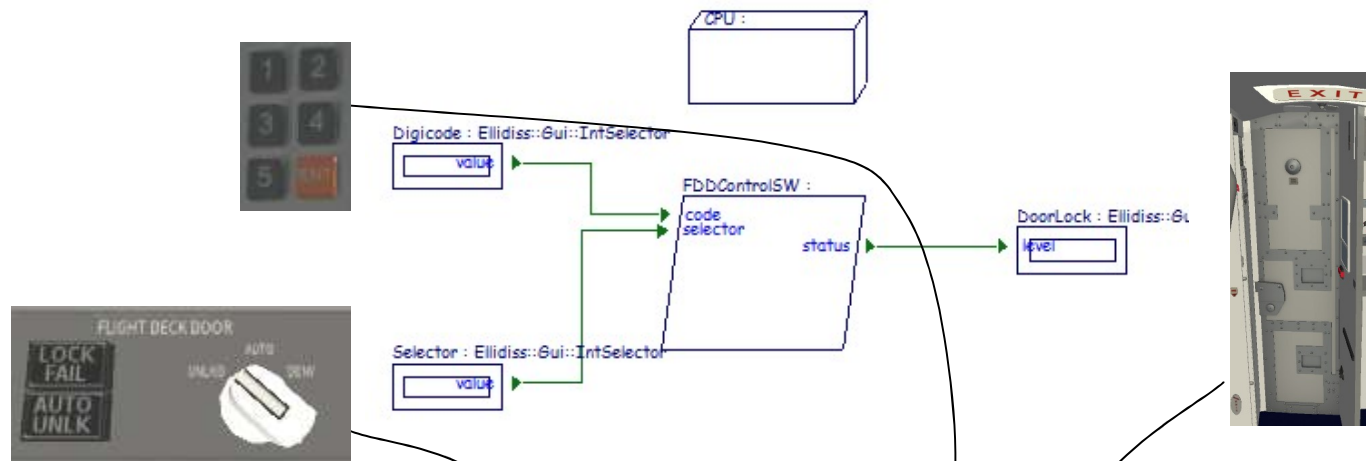


Exemple: distributed system timing analysis

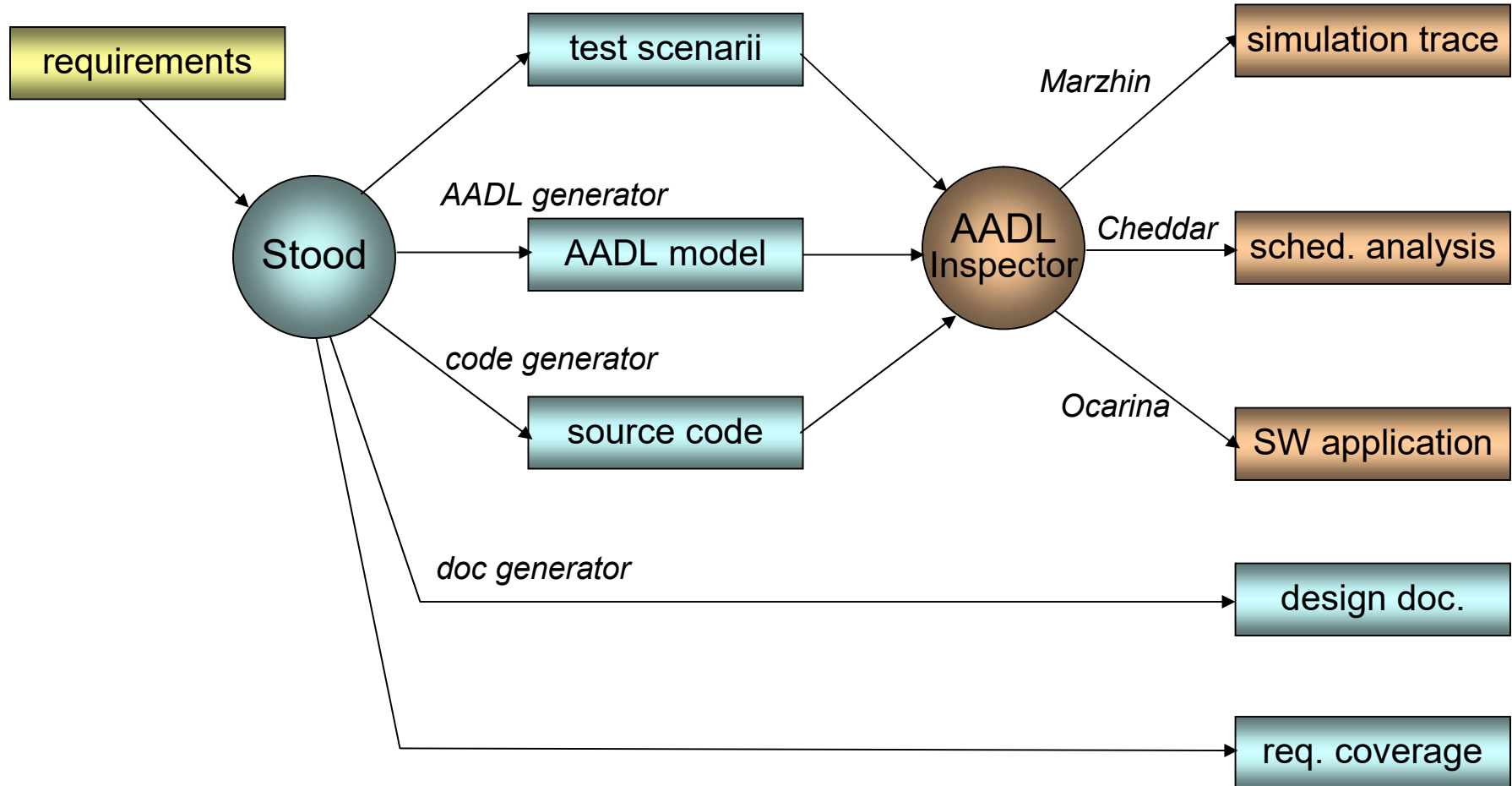


Human in the Loop

Replace the GUI dialog box by 3D interactive widgets to animate a complete virtual reality scene



Integrated Workflow



Case study: Pacemaker

1. Requirements (ref. Brian Larson)

The "s" port connects to an analog front-end which filters signals from leads inside the heart. When the signal exceeds a threshold (~ 3 mV) the front-end sends a "sense" event to the VVI thread.

The "p" port sends an event to the front end to issue a "pace"--a half-millisecond long, ~ 2 V--to the heart to induce contraction.

The behavior is:

- *when the heart is beating fast enough, do nothing.*
- *when the heart has not had a beat for 1000 ms (lrl), cause a pace*
- *if the sense comes too soon after a beat, < 300 ms (vrp), ignore it.*

The "n" sends an event (ignored here) when the signal from the front end was not during the ventricular refractory period (vrp), so is regarded as a true heart beat.

2. Test cases (ref. Brian Larson)

Test 1) No sensing.

The thread will put out an event on the "p" port every 1000 ms.

Test 2) Normal rhythm.

Put an event on the "s" port every 900 ms. The thread will put an event out the "n" port each dispatch.

Test 3) Ignore sense in VRP.

Wait 1000 ms for the first pace; 200 ms later put an event on the "s" port.

The next pace will occur at 2000 ms.

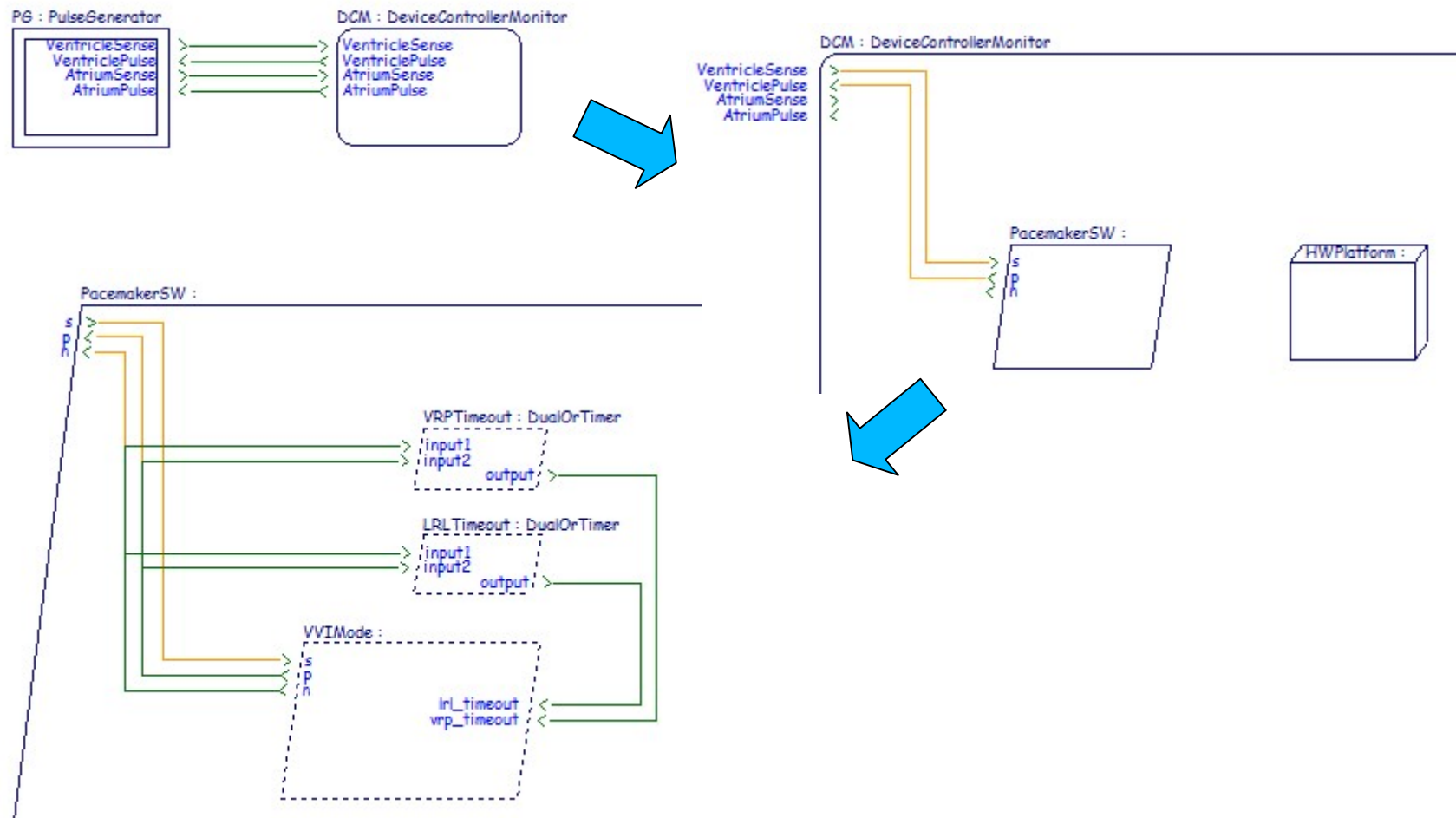
Test 4) Pace after sense.

Wait 1000 ms for the first pace; 200 ms later put an event on the "s" port, which will be ignored.

At 1400 ms put out another event on the "s" port. Expect the next pace at 2400 ms.

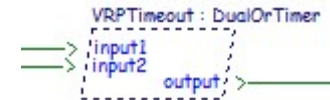
Case study: Pacemaker

3. Architectural Design with Stood



Case study: Pacemaker

4. AADL code generated by Stood



```

THREAD DualOrTimer
FEATURES
  input1 : IN EVENT PORT;
  input2 : IN EVENT PORT;
  output : OUT EVENT PORT;
PROPERTIES
  Dispatch_Protocol => Timed;
ANNEX Behavior_Specification {**
  STATES
    s1 : INITIAL COMPLETE FINAL STATE;
  TRANSITIONS
    t1 : s1 -[ ON DISPATCH input1 ]-> s1;
    t2 : s1 -[ ON DISPATCH input2 ]-> s1;
    t3 : s1 -[ ON DISPATCH TIMEOUT ]-> s1 { output! };
  **};
END DualOrTimer;

```


Case study: Pacemaker



THREAD VVIMode

FEATURES

```

s : IN EVENT PORT;
p : OUT EVENT PORT;
n : OUT EVENT PORT;
lrl_timeout : IN EVENT PORT;
vrp_timeout : IN EVENT PORT;
END VVIMode;

```

THREAD IMPLEMENTATION VVIMode.others

SUBCOMPONENTS

```
vrp : DATA int;
```

PROPERTIES

```
Dispatch_Protocol => Aperiodic;
```

ANNEX Behavior_Specification {**

STATES

```
s1 : INITIAL COMPLETE FINAL STATE;
```

TRANSITIONS

```
t0 : s1 -[ ON DISPATCH vrp_timeout ]-> s1
    { vrp := 0 };
```

```
t1 : s1 -[ ON DISPATCH s ]-> s1
    { if (vrp = 0) n!; vrp := 1 end if };
```

```
t2 : s1 -[ ON DISPATCH lrl_timeout ]-> s1
    { p!; vrp := 1 };
```

```
**};
```

END VVIMode.others;

Case study: Pacemaker

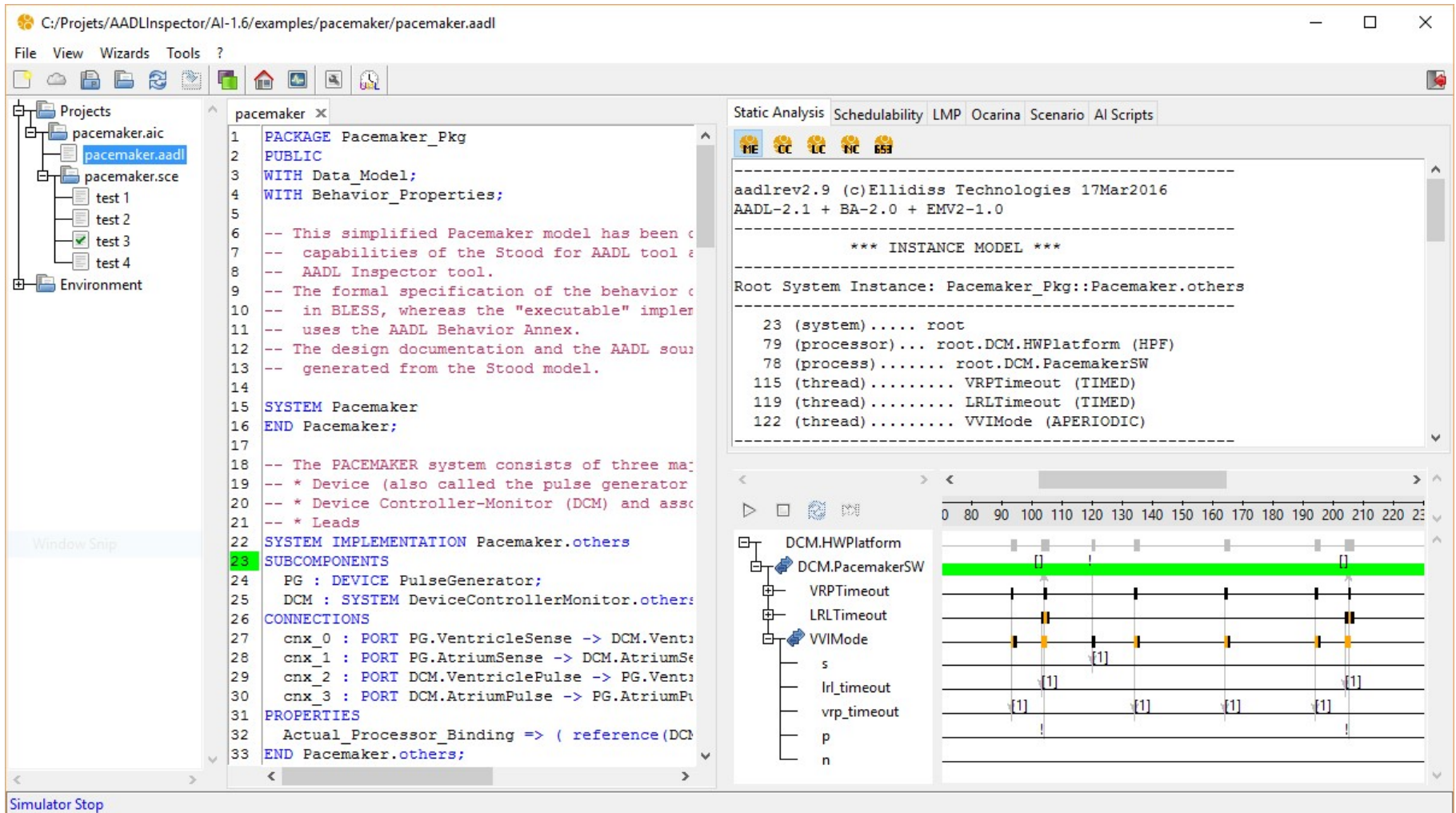
instance model
identifier

5. Test case scenarii

```
<scenarii>
  <interface>
    <event lmpId="sense"
      id="dcm.hwplatform.dcm.pacemakersw.vvimode.s"/>
  </interface>
  <scenario name="test 1" description="No sensing">
  </scenario>
  <scenario name="test 2" description="Normal rhythm">
    <action nextTick="tick+90" tick="0" value="1" >
      <feature lmpRef="sense"/>
    </action>
  </scenario>
  <scenario name="test 3" description="Ignore sense in VRP">
    <action tick="120" value="1">
      <feature lmpRef="sense"/>
    </action>
  </scenario>
  <scenario name="test 4" description="Pace after sense">
    <action tick="120" value="1">
      <feature lmpRef="sense"/>
    </action>
    <action tick="140" value="1">
      <feature lmpRef="sense"/>
    </action>
  </scenario>
</scenarii>
```

Case study: Pacemaker

6. Test case execution with AADL Inspector



The screenshot displays the AADL Inspector application window, showing the execution of a test case for the Pacemaker system. The interface is divided into several panes:

- Left Pane (Projects):** Shows a tree view of the project structure, including 'pacemaker.aic', 'pacemaker.aadl', 'pacemaker.sce', and four test cases (test 1, test 2, test 3, test 4). 'test 3' is selected.
- Center Pane (Code Editor):** Displays the AADL source code for 'Pacemaker_Pkg'. The code includes package declarations, system definitions, and implementation details for the 'Pacemaker.others' system.
- Right Pane (Static Analysis):** Shows the results of the static analysis, including the AADL version (2.9), the tool version (17Mar2016), and the instance model. The instance model lists the root system and its components, such as 'root.DCM.HWPlatform (HPF)', 'root.DCM.PacemakerSW', and various threads like 'VRPTimeout (TIMED)', 'LRLTimeout (TIMED)', and 'VVMMode (APERIODIC)'.
- Bottom Pane (Timeline):** Displays a timeline of the test case execution. The timeline shows the execution of the 'DCM.HWPlatform' and 'DCM.PacemakerSW' components, with various events and messages (e.g., 'VRPTimeout', 'LRLTimeout', 'VVMMode') occurring over time. The timeline is marked with a scale from 0 to 230.

The status bar at the bottom indicates 'Simulator Stop'.

7. Design documentation



AADL design documentation

Generated by Stood

Table of contents

1.	PROJECT	3
1.1.	Design Tree	3
1.2.	AADL Diagram	3
2.	SYSTEM Pacemaker IS	4
2.1.	DESCRIPTION	4
2.1.1.	PROBLEM	4
2.1.2.	SOLUTION	4
2.1.3.	PROPERTIES	5
2.2.	IMPLEMENTATION	5
2.2.1.	SUBCOMPONENTS	5
2.2.2.	BEHAVIOR	6
3.	DEVICE PG IS	6
3.1.	DESCRIPTION	6
3.1.1.	PROBLEM	6
3.1.2.	SOLUTION	7

Conclusion

- **COTS tools**
 - Stood for HOOD: software design and coding (Ada, C)
 - Stood for AADL: instance model graphical editor for AADL
 - AADL Inspector: model processing framework:
 - Rules checkers
 - Scheduling analysis
 - Simulation
- **Technology**
 - LMP: model processing toolbox
 - GMP: domain specific graphical editor framework
- **Services**
 - Commercial support
 - Graphical front ends development
 - Model processing tools (rules checkers, generators)
 - Model transformations
 - Heterogenous tools integration
 - R&D partnerships