Stood for AADL

AADL committee
Tucson, 24 January 2007

Ellidiss Software
www.ellidiss.com

Stood overview

- **Modelling process:**
  - modelling methodology (projects, designs):
    - AADL System instance: HW+SW specification
    - AADL Process instance: from SW architecture to code
    - AADL Packages: libraries of Data component
  - AADL graphical editor
  - import: textual AADL; legacy Ada or C code
  - industrial features:
    - concepts of Project and Design
    - requirements traceability
    - interface with conf/version management
    - distributed development

- **Model verification:**
  - bidirectional interface with Topcased/Osate
  - embedded legality rules checker
  - schedulability analysis: connection with Cheddar

- **Production:**
  - textual AADL generator
  - Ada, C and C++ code generators
  - round-trip engineering
  - design report generator

- **Current distribution: v5.1**
  - download: www.ellidiss.com
  - support: stood@ellidiss.com
  - available on Windows, Linux and Solaris
  - light weight: 25 Mb disk space

Ellidiss Software
www.ellidiss.com

Example

code and document generators

AADL graphical editor

project structure

component structure

Stood for AADL - demo-aadl

File Edit Design Component Feature Tools Help

(design) my_system

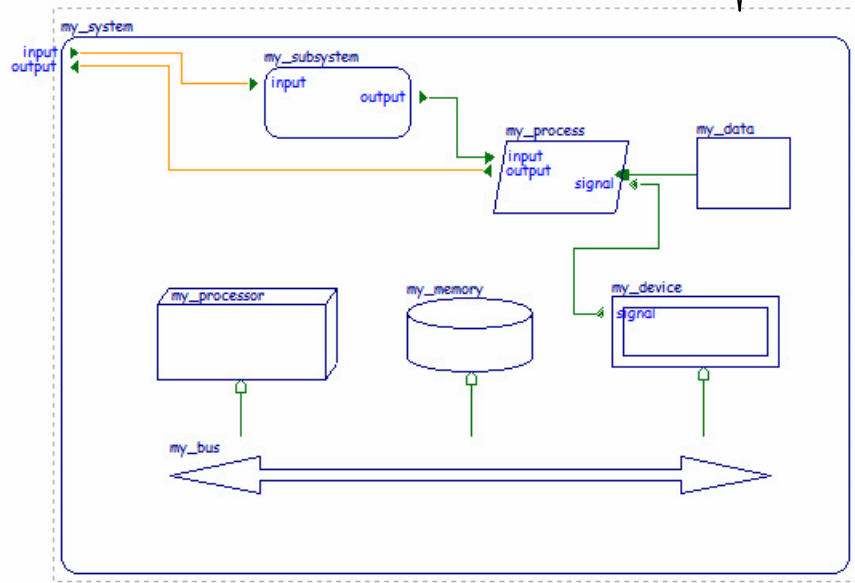
- (design) my_package
  - my_package
    - Data1
    - Data2
    - Data3
- (design) my_process
  - my_process
    - Thread1
    - Thread2
    - my_thread_group
      - my_process11
    - my_data
    - my_package
- (design) my_system
  - my_system
    - my_process
    - my_processor

ods | ada | c | cpp | aadl | test | checks

- PACKAGE or COMPONENT
  - DESCRIPTION
    - PROBLEM
      - Statement of the Problem (text)
      - Sketch of the Problem
      - Referenced Documents
    - SOLUTION
  - PROVIDED INTERFACE
    - PROPERTIES
    - PORT GROUPS
    - PORTS and SUBPROGRAMS
      - input
      - output
  - REQUIRED INTERFACE
  - DATAFLOWS
  - IMPLEMENTATION

Requirements | Graphic Design | Detailed Design | Checkers | Code | Documentation | Deployment

Hood
UML
AADL



Save text | Statement of the Problem (text)

This AADL component describes a complete system, including software and execution platform subcomponents.

Ellidiss Software
www.ellidiss.com

R&D contribution

- **ASSERT:**

- promoting the AADL to support the ASSERT modeling process:
  - Data view: Packages and Data from ASN.1 declarations
  - Functional view: Subprogram components and calls
  - Interface view: Systems and subprogram features
  - Concurrency view: Processes, Threads and Data instances
  - Deployment view: Virtual Processors, ... (in progress)
- a Technical Note is in progress
- thanks to the AADL, ASSERT Pilot Projects may use either Stood or Topcased

- **SMP2:**

- Simulation Model Portability: European Space Agency standard
- specifies a simulation runtime API, modeling and C++ coding guidelines
- SMP2 prototyping study:
  - context: French Space Agency (CNES) R&T program
  - analogy between managed simulations and scheduled real-time applications
  - ⇒ mapping between SMP2 and AADL concepts
  - ⇒ implementation of a SMP2 compliant C++ code generator in Stood
- first industrial experiments with Spacebel (Belgium)
- other industrial experiments are foreseen (Astrium, Alcatel,...)

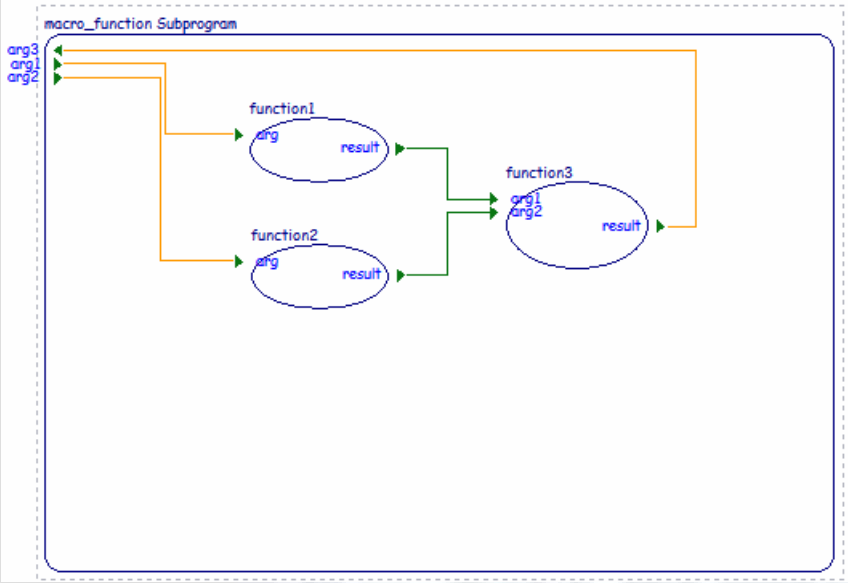
- **TOPCASED/ET (External Tools):**

- Registering Stood and Cheddar as remote service providers on the Topcased bus
- Defining service chains

Ellidiss Software
www.ellidiss.com

new in Stood 5.2

- **Functional modelling:**
  - graphical representation of subprogram calls and parameter connections (IDEF-0 like)
  - generation of corresponding AADL code.




```
PACKAGE my_functions
PUBLIC

  SUBPROGRAM macro_function
  FEATURES
    arg3 : OUT PARAMETER T_Flow;
    arg1 : IN PARAMETER T_Flow;
    arg2 : IN PARAMETER T_Flow;
  END macro_function;

  SUBPROGRAM IMPLEMENTATION macro_function.others
  CALLS {
    function1 : SUBPROGRAM my_functions::function;
    function2 : SUBPROGRAM my_functions::function;
    function3 : SUBPROGRAM my_functions::function3;
  };
  CONNECTIONS
    PARAMETER arg1 -> function1.arg;
    PARAMETER arg2 -> function2.arg;
    PARAMETER function3.result -> arg3;
    PARAMETER function1.result -> function3.arg1;
    PARAMETER function2.result -> function3.arg2;
  END macro_function.others;

  SUBPROGRAM function
  FEATURES
    arg : IN PARAMETER T_Flow;
    result : OUT PARAMETER T_Flow;
  END function;

  SUBPROGRAM function3
  FEATURES
    arg1 : IN PARAMETER T_Flow;
    arg2 : IN PARAMETER T_Flow;
    result : OUT PARAMETER T_Flow;
  END function3;

END my_functions;
```

- **Release:**
  - planned end January 2007