Architecture Analysis & Design Language

Tutorial

Jean-François Tilman
Jean-Francois.Tilman@axlog.fr
Jean-Francois.Tilman@laposte.net

Axlog Ingénierie

2007-09-18

Outline

Outline

Definition

- AADL = Architecture Analysis & Design Language
- A precise, complete and non-ambiguous language to describe embedded systems (software and execution platform)
- A textual and graphical format
- An international standard

Objectives

- Having a means to formally describe a system for:
  - Design of the system
  - Analysis (schedulability, dimensionning, performances, safety, etc.)
  - Use of proofs of properties
  - Automatic code generation...
- Using a single description during the whole development life cycle
  - Seamless development process
  - Improvement of communication between actors

Application Fields

Any domain using complex, real time, critical embedded systems:

- Avionics and space
- Automotive, railway, transportation
- Robotics
- Nuclear industry
- ...

History of AADL

1991	MetaH developed by Honeywell
2000	Beginning of the standardization of AADL (“Avionics architecture description language”)
2003	Name changed into “Architecture Analysis & Design Language”
2004	Publication of version 1 *(ref. SAE AS5506)*
2005	Publication of annexes to the standard
2007 ?	AADL version 2

Outline

Standardization committee

International committee, mainly composed of American and European members from avionics, defense and automotive domains

Under authority of the Society of Automotive Engineers (SAE)

- Avionics system division (ASD)
  - Embedded system committee (AS2)
    - AADL subcommittee (AS2C)

Contact : Bruce Lewis, AS2C chairman

Reasons for standardization

Why a standard?

- Ensures a common understanding of the language and avoids divergences
- Put together several contributions and experiences

Why an international standard?

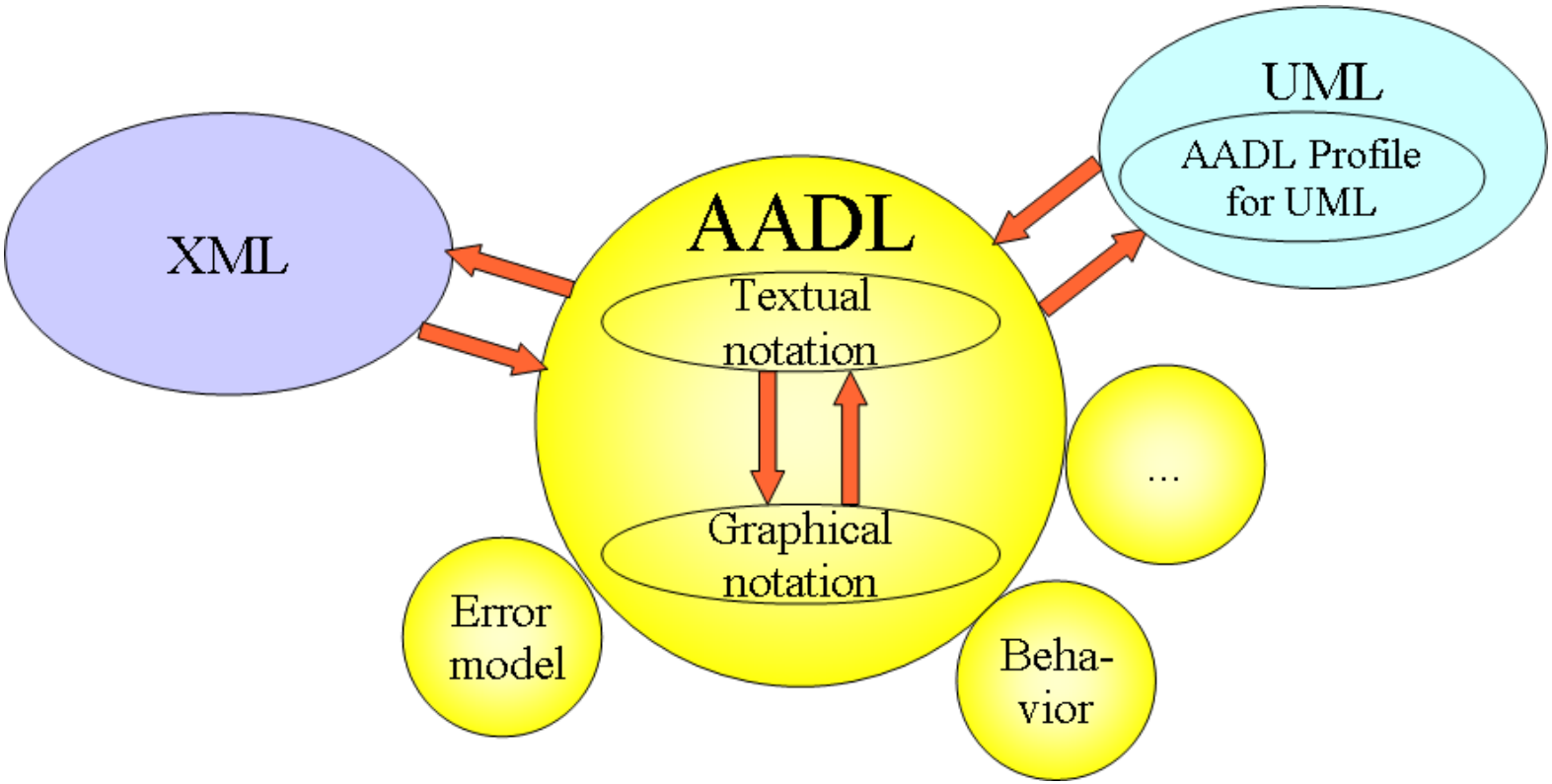
- Larger adoption by industry
- Industrial exchanges are nowadays international
- Take opportunity of various approaches and sensibilities

Outline

Principles of an AADL description

- An AADL architecture is a set of components which are hierarchically composed
- The interface of a component is represented by a set of features
- The components are connected through their interfaces
- Additional mechanisms refine the description

AADL Positioning



Iterative approach

AADL makes possible iterative development processes

- Refinement mechanisms
- Valid incomplete descriptions

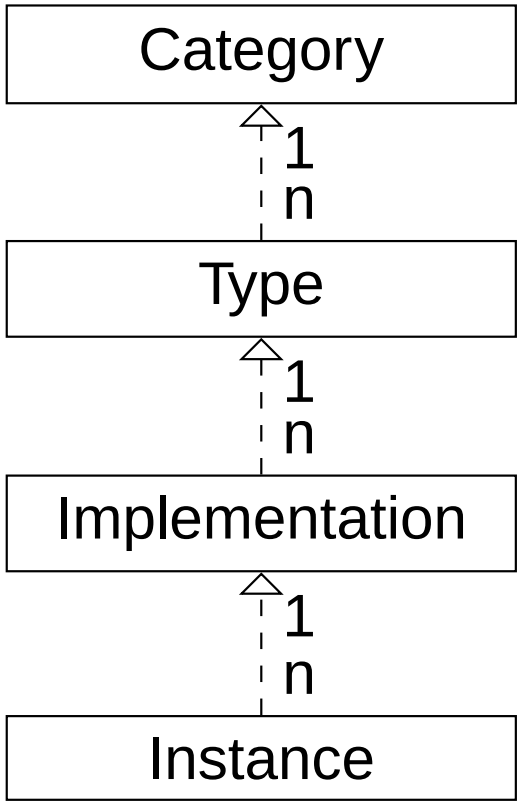
Interests:

- Collaborative work
- Libraries of reuseable components

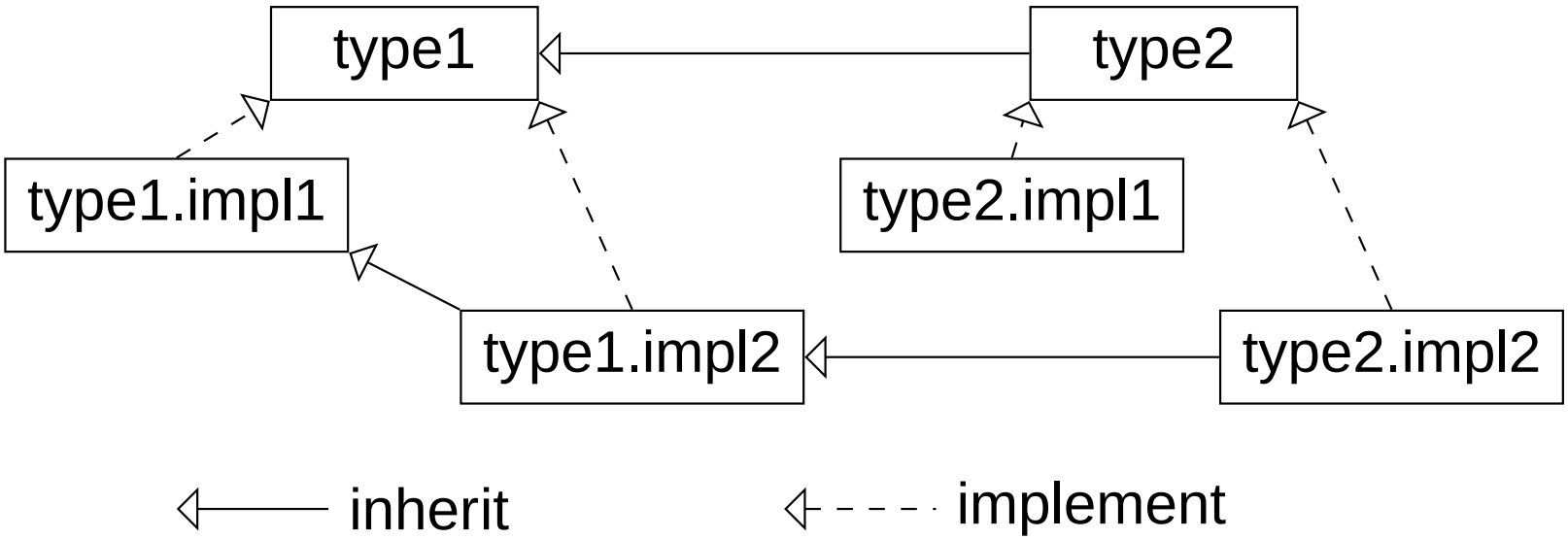
Outline

Component Description Levels

- The category (predefined) gives the semantics
- The type describes the external interface
- The implementation describes the contents
- The instance is a realisation of a type or implementation



Inheritance



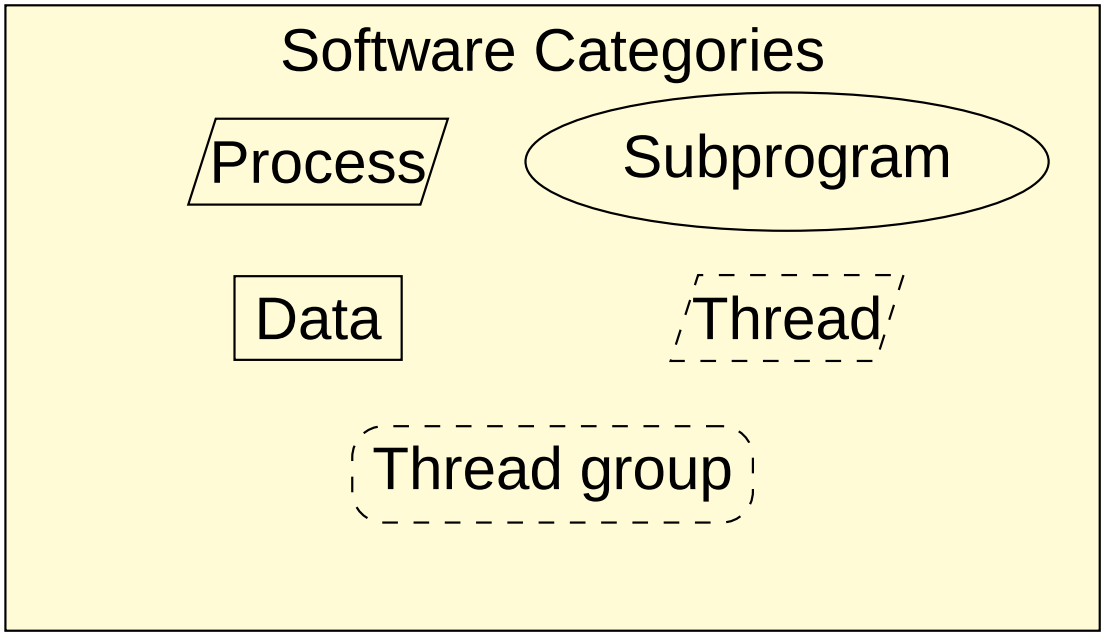
- A type can extend another type
- An implementation can extend another implementation
- The type/implementation relations are combined with the inheritance relations

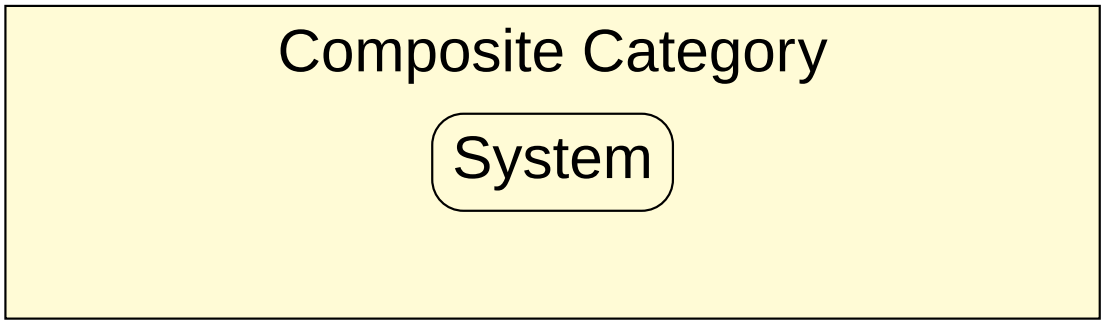
Component Categories (1/2)

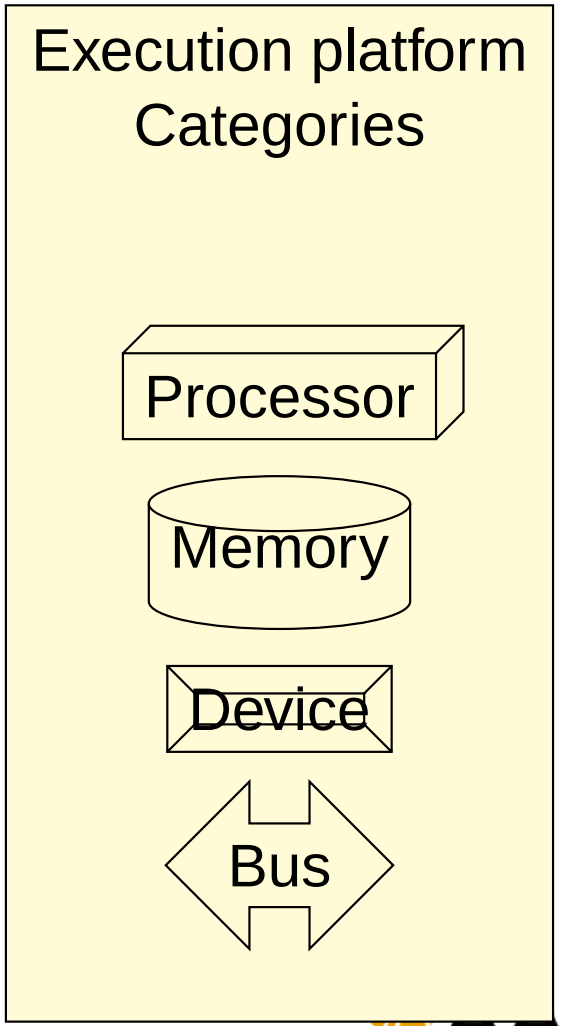
Each component category

- defines a strong specific semantics
- restrains the contents of its components
- defines a set of standard properties

Component Categories (2/2)







Component Types

- Specifies the external interface of a component, that its implementations satisfy
- Contains:
  - feature declarations
  - property associations
  - annex clauses

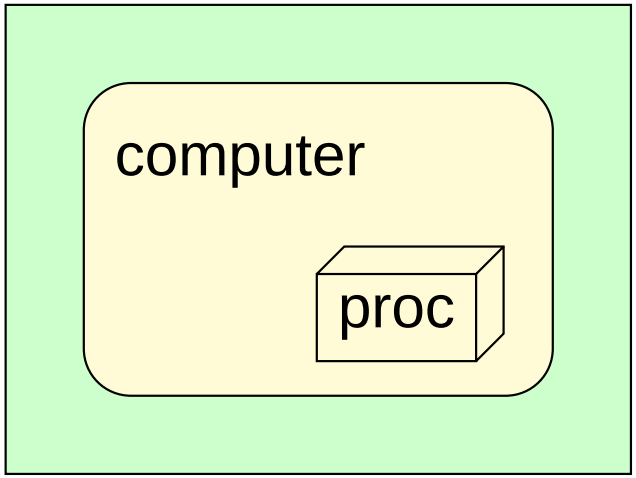
Component Implementations

- The implementation of a component must be conform to the corresponding AADL type
- The implementation of a component may contain:
  - subcomponent declarations
  - connection between features
  - property associations
  - mode declarations
  - calls of subprograms
  - annex clauses

Subcomponents

A subcomponent is:

- An instance of a component
- Contained into a component implementation
- Defined by:
  - its category (mandatory)
  - its type (optional)
  - its implementation (optional)



Properties

- Properties bring additional information on AADL elements
- A predefined set of properties is specified by the standard

Example

```
thread implementation T1.impl
  properties
    Period => 120 Ms ;
    Compute_Execution_Time => 30 Ms .. 40 Ms ;
end T1.impl ;
```

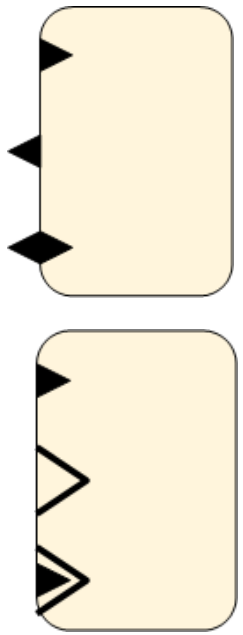
Outline

Feature Categories

- A feature specifies how a component interfaces with other components in the system
- Kinds of features:
  - Port
  - Parameter
  - Subcomponent access

Port

- A port is a logical connection point that can be used for the transfer of control and data between components
- Three directions:
  - input (in)
  - ouput (out)
  - bidirectional (in out)
- Three types of port:
  - data
  - event
  - event data



Parameter

- A subprogram parameter represents a data value which pass into and out of subprograms
- It is typed with a data component type
- Three directions:
  - input (in)
  - output (out)
  - bidirectional (in out)

Subcomponent Access

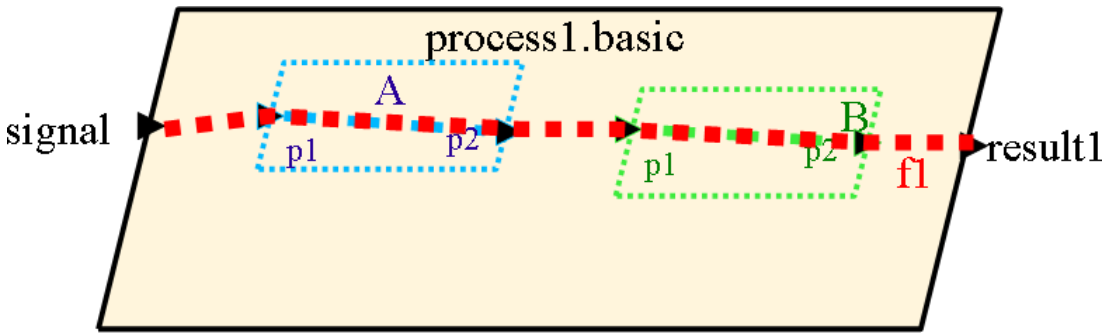
- A subcomponent (*data* or *bus*) can be made available from the outside of its hierarchical container (provided access)
- A component may declare it requires access to an external subcomponent (required access)

Connection

- A connection represents a link between component features
- Three types of connections:
  - Port connection which represents
    - a transfer of data and control
  - Parameter connection which represents
    - a data flow between the parameters of a sequence of subprogram calls
  - Access connection which represents
    - the communication between hardware components through a shared bus
    - the access to a shared *data* component by a software component

Outline

Flow

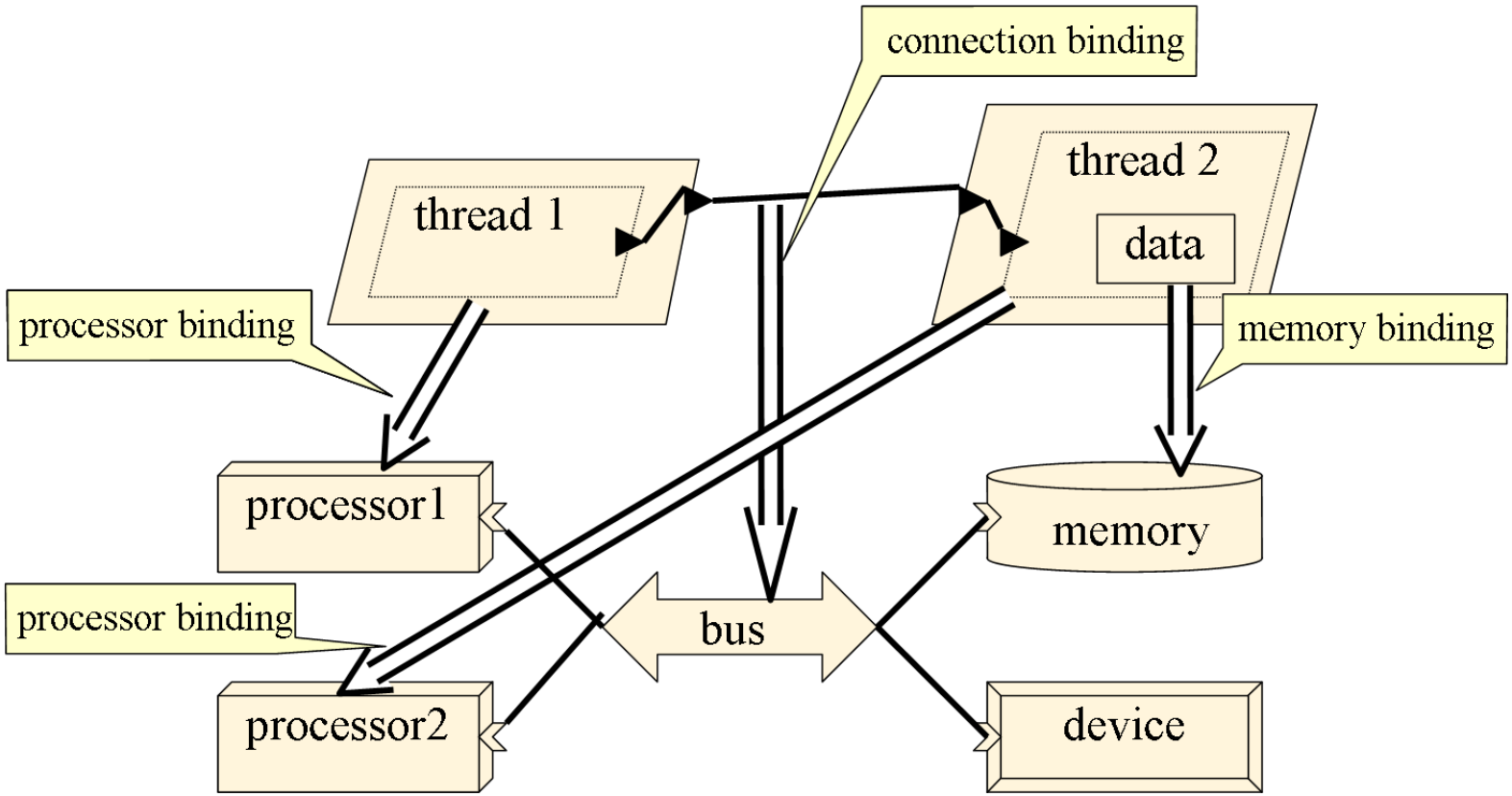


- Purpose: supporting flow analyses
  - Timing
  - Latency
  - Reliability
  - Quality of service
  - ...

Bindings

- Software components may be bound onto hardware components:
  - threads onto processors
  - processes and data onto memories
  - connections onto buses
- Binding properties:
  - Restriction on allowed binding
    - `Allowed_Processor_Binding`
    - `Allowed_Memory_Binding`
    - `Allowed_Connection_Binding`
  - Actual binding
    - `Actual_Processor_Binding`
    - `Actual_Memory_Binding`
    - `Actual_Connection_Binding`

Example of Bindings



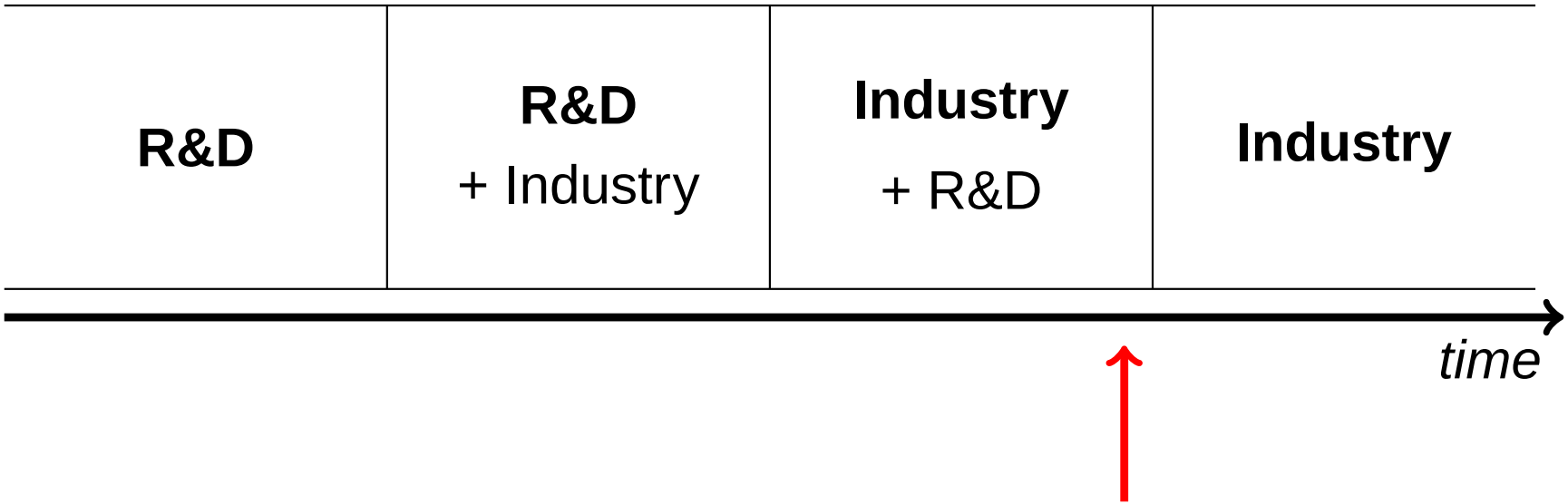
Modes

- A mode represents the operational state of a system
- Mode transistions model a dynamic behavior (which consists of configurations switches)
- At one time, a component is in exactly one mode
- A component may have:
  - property values which are mode-dependant
  - subcomponents or connections which are mode-dependant
- A mode transition is triggered by an event
- System Operational Mode is a combination of local modes

Outline

Adoption by Industry

- AADL is going to be used for real industrial projects



Impact of AADL in industry and research

- Used by: Airbus, Axlog, Boeing, Bosch, Dassault Aviation, EADS, Ellidiss, ESA, Ford, General Dynamics, Honeywell, Lockheed Martin, Rockwell Collins, Sagem, Toyota...
- Used in: aerospace, avionics, automotive, unmanned vehicles, medical...
- Impact through standards: OMG (UML2 profile), Autosar, ARINC653, NATO, SAE...

Source: Peter Feiler,

`http://www.aadl.info/documents/SEIAADLMBEImpact42007-phf.pdf`

Outline

Interest of AADL extensions

The potential users of AADL have specific and different needs

- The expression capability of the language has to be extensible

An AADL description has to remain understandable by everyone

- Syntax and precise semantics of the language must be respected

AADL proposes extension mechanisms which meet these two requirements

User Property Sets

- The user may define new AADL properties and associate them to any AADL element

Example

```
property set mass_properties is
  Mass_Type : type aadlreal
              units ( g, kg => g * 1000 ) ;
  Mass : mass_properties::Mass_Type
         applies to ( bus, device,
                      memory, processor ) ;
end mass_properties ;
```

Language Annexes

- An annex is a block of code in a user sublanguage
- It may be ignored by anyone not knowing it

Example

```
thread Collect_Sample
  features
    Input_Sample : in data port SampleData;
    Output_Average : out data port SampleData;
  annex OCL {**
    pre: 0 < Input_Sample < maxValue;
    post: 0 < Output_Sample < maxValue;
  **};
end Collect_Samples;
```

Standard Annexes

- Language annexes may be standardized by the committee
- They extend the description capabilities without complexifying the core language
- Annexes currently standardized
  - Error model annex
  - Behavior annex

Collaboration with the Committee

- The AADL committee is ready to consider suggestions and contributions
- A way to extend AADL capabilities for
  - general purpose issues
  - interesting the whole AADL community
  - hardly solved by other solutions
- Most of the AADL v2 changes come from such collaborations

Outline

Tool Support

Tools supporting AADL now exist

- Tools specially developed for AADL
- AADL import/export functions for existing tools

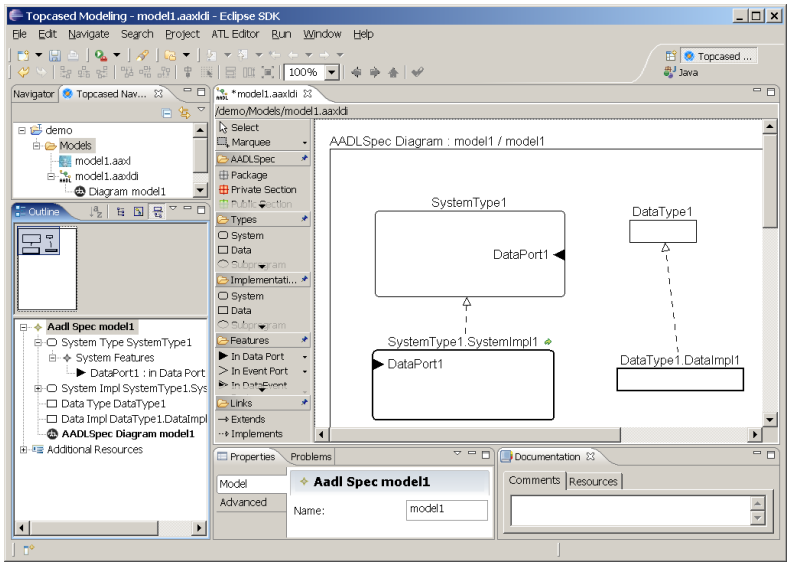
Some tools for:

- Modeling
- Verification and validation
- Code generation
- ...

Osate

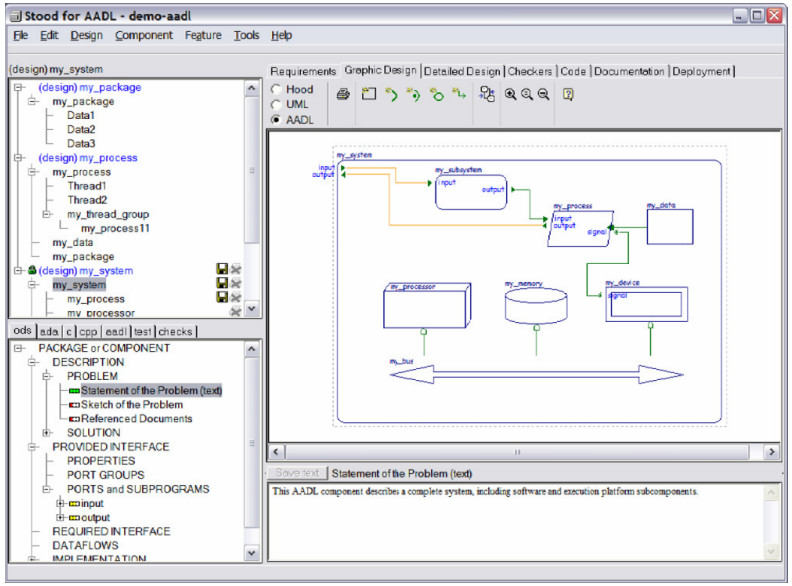
- Open source AADL tool environment
- Purpose: Modeling systems with AADL
- Features:
  - Eclipse based AADL textual editor
  - Framework to develop and integrate other AADL tools
- http://www.aadl.info/tool/osate.html

Topcased



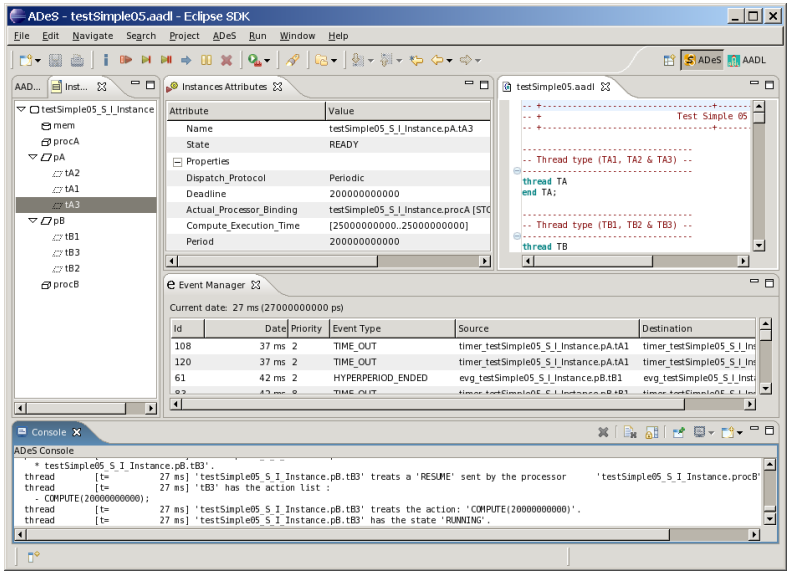
- Graphical modeling tool on top of Osate
- Purpose: Modeling systems with AADL
- http://www.topcased.org

Stood



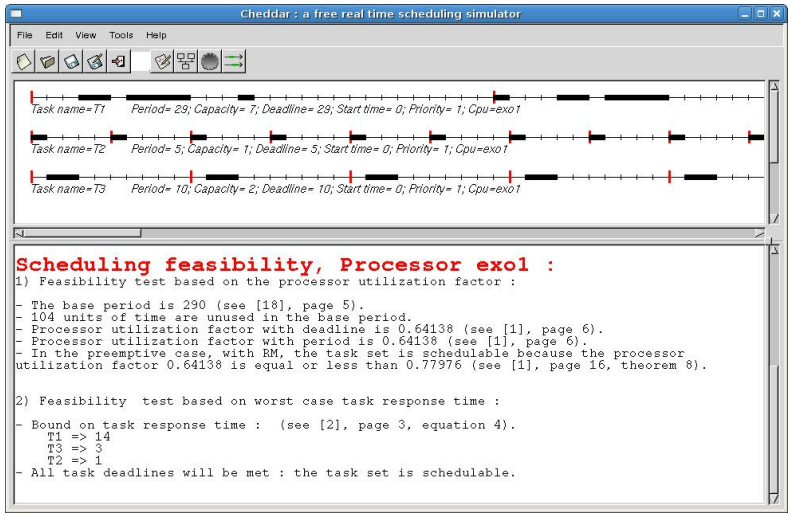
- Purpose: Embedded software development with UML, HOOD and AADL
- http://www.ellidiss.com/stood.shtml

ADeS



- Purpose: Simulating the behavior of AADL architectures
- http://www.axlog.fr/aadl/ades_en.html

Cheddar



- Purpose: Real time scheduling analysis
- http://beru.univ-brest.fr/∼singhoff/cheddar/

Other tools

Ocarina	AADL $\rightarrow$ X generator http://ocarina.enst.fr/
Furness Toolset	Collection of open source AADL tools http://www.furnesstoolset.com/
MetaH	The ancestor of AADL and its toolset http://www.htc.honeywell.com/metah/

...

AADL courses and support

- AADL presented in workshops, conferences, tutorials
- Training sessions available
  - SEI (http://www.sei.cmu.edu/products/courses/p52.html)
  - Adalog (http://www.adalog.fr/aadlf2.htm)
  - Pyrrhus Software (http://www.pyrrhusoft.com/)
- Commercial support and expertise proposed by companies

Conclusion

- AADL is dedicated to the development of complex systems
- It is considered by a larger and larger industrial community
- It may be extended to cover particular needs
- Support and tools exist
- $\Rightarrow$ Just try it!

More information

- AADL official web site: `http://www.aadl.info`
- AADL standard: `http://www.sae.org/technical/standards/AS5506`
- Contact: `aadl@axlog.fr`