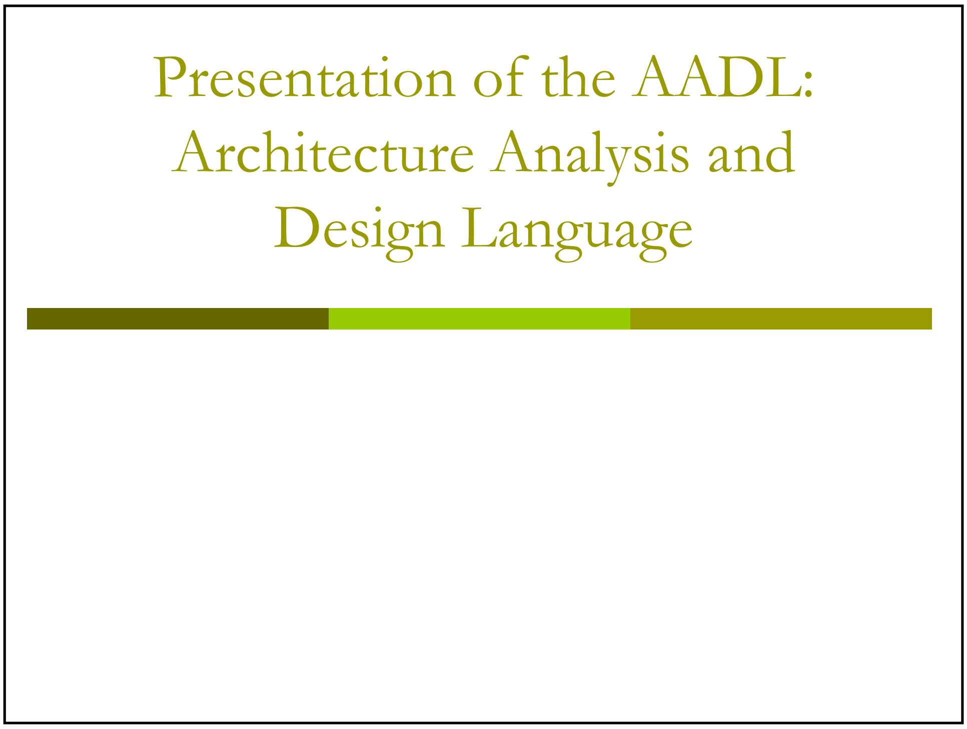
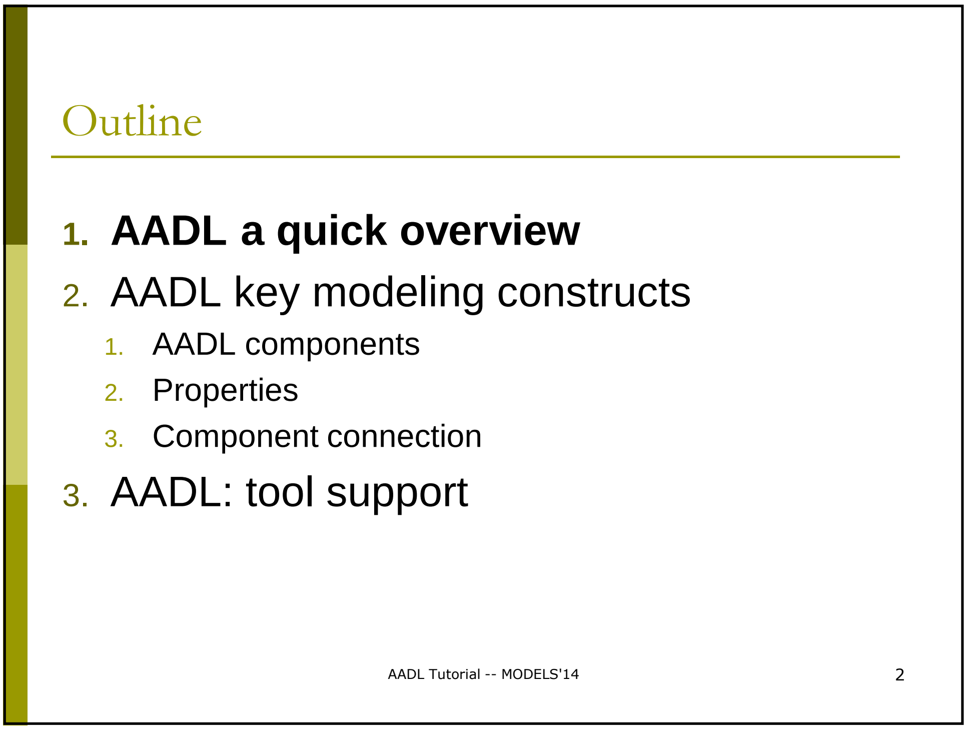
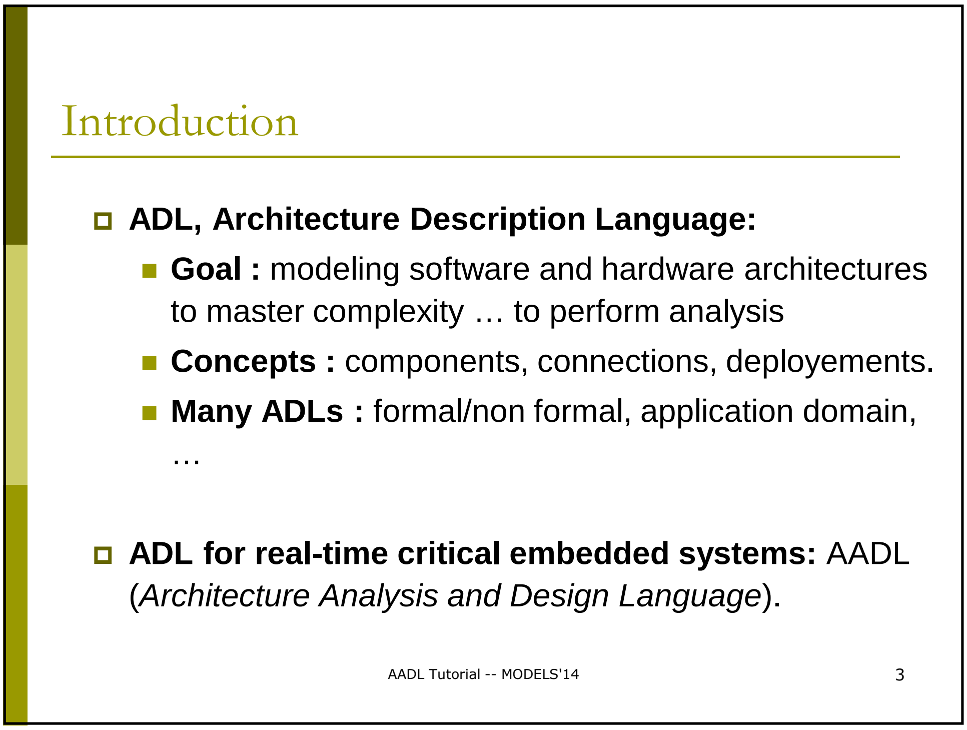


Presentation of the AADL: Architecture Analysis and Design Language



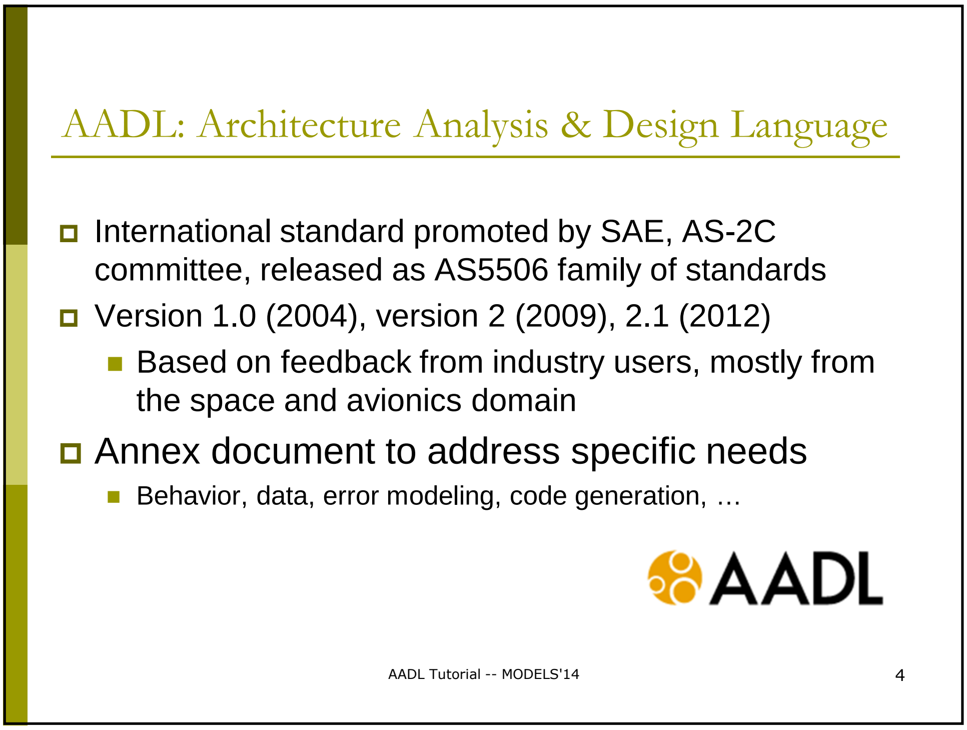
Outline

1. **AADL a quick overview**
2. AADL key modeling constructs
   1. AADL components
   2. Properties
   3. Component connection
3. AADL: tool support



Introduction

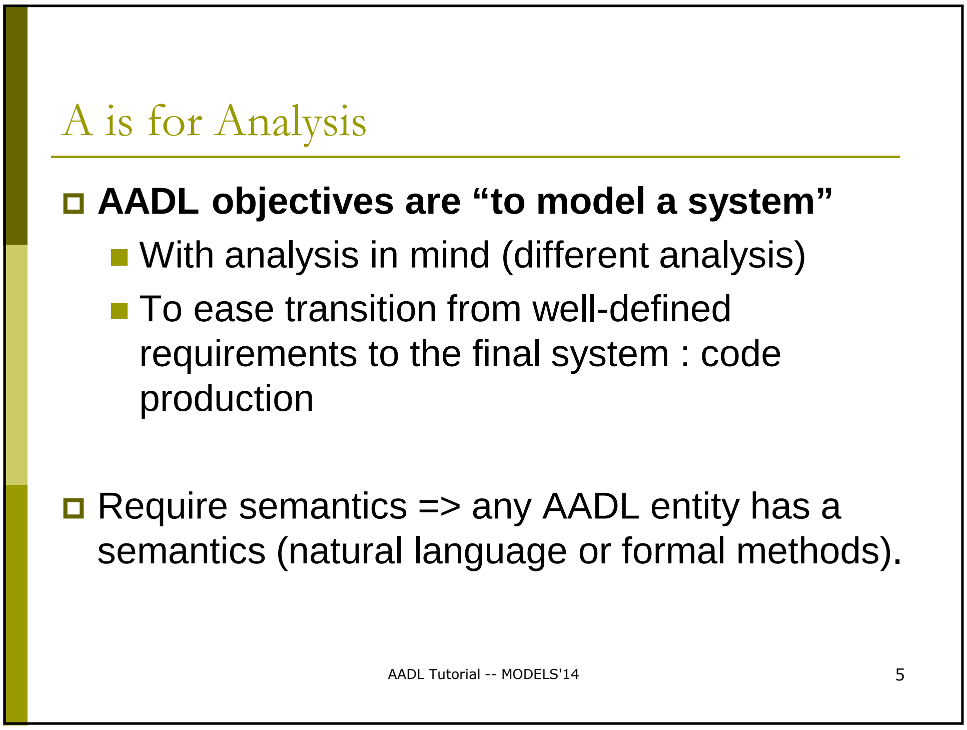
- **ADL, Architecture Description Language:**
  - **Goal :** modeling software and hardware architectures to master complexity … to perform analysis
  - **Concepts :** components, connections, deployements.
  - **Many ADLs :** formal/non formal, application domain, …
- **ADL for real-time critical embedded systems:** AADL (*Architecture Analysis and Design Language*).



AADL: Architecture Analysis & Design Language

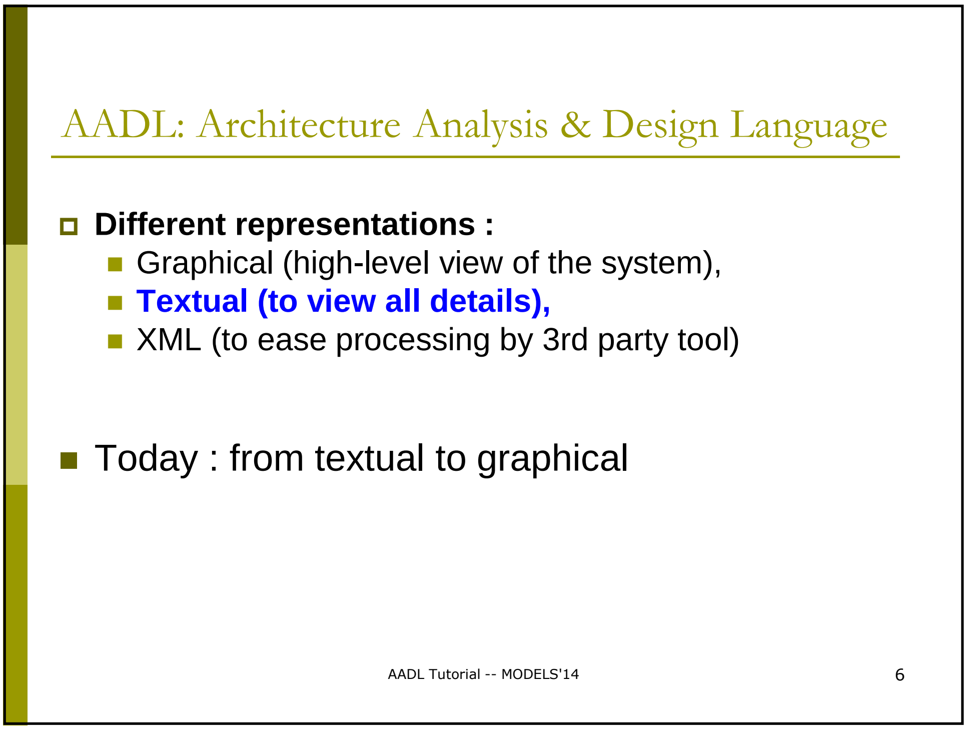
- International standard promoted by SAE, AS-2C committee, released as AS5506 family of standards
- Version 1.0 (2004), version 2 (2009), 2.1 (2012)
  - Based on feedback from industry users, mostly from the space and avionics domain
- Annex document to address specific needs
  - Behavior, data, error modeling, code generation, …

AADL



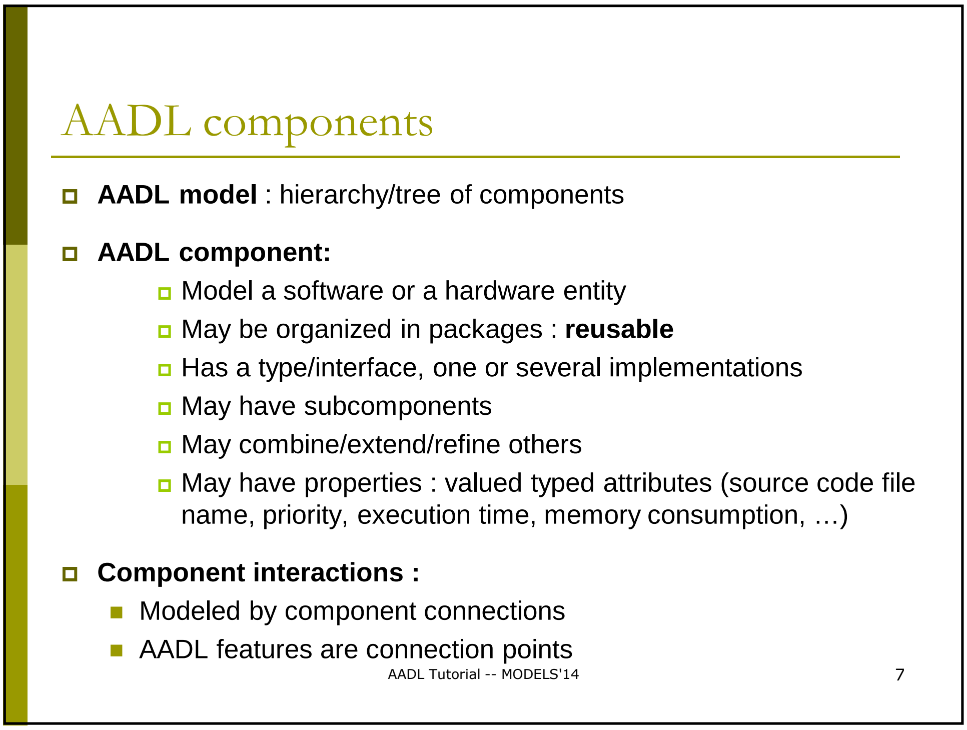
A is for Analysis

- **AADL objectives are “to model a system”**
  - With analysis in mind (different analysis)
  - To ease transition from well-defined requirements to the final system : code production
- Require semantics => any AADL entity has a semantics (natural language or formal methods).



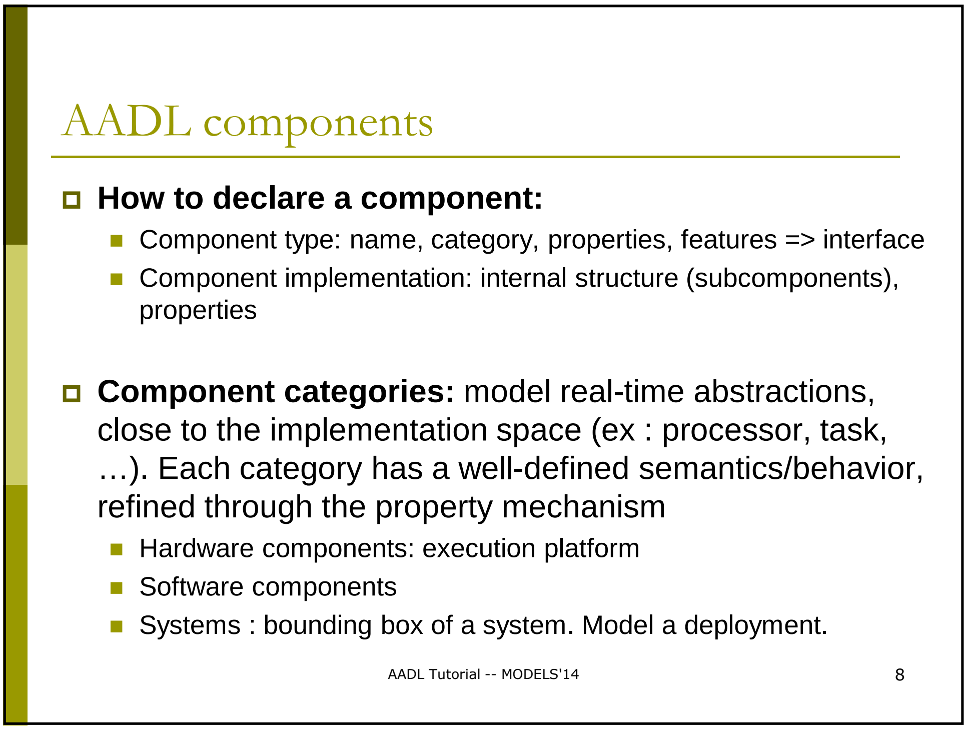
AADL: Architecture Analysis & Design Language

- **Different representations :**
  - Graphical (high-level view of the system),
  - **Textual (to view all details),**
  - XML (to ease processing by 3rd party tool)
- Today : from textual to graphical



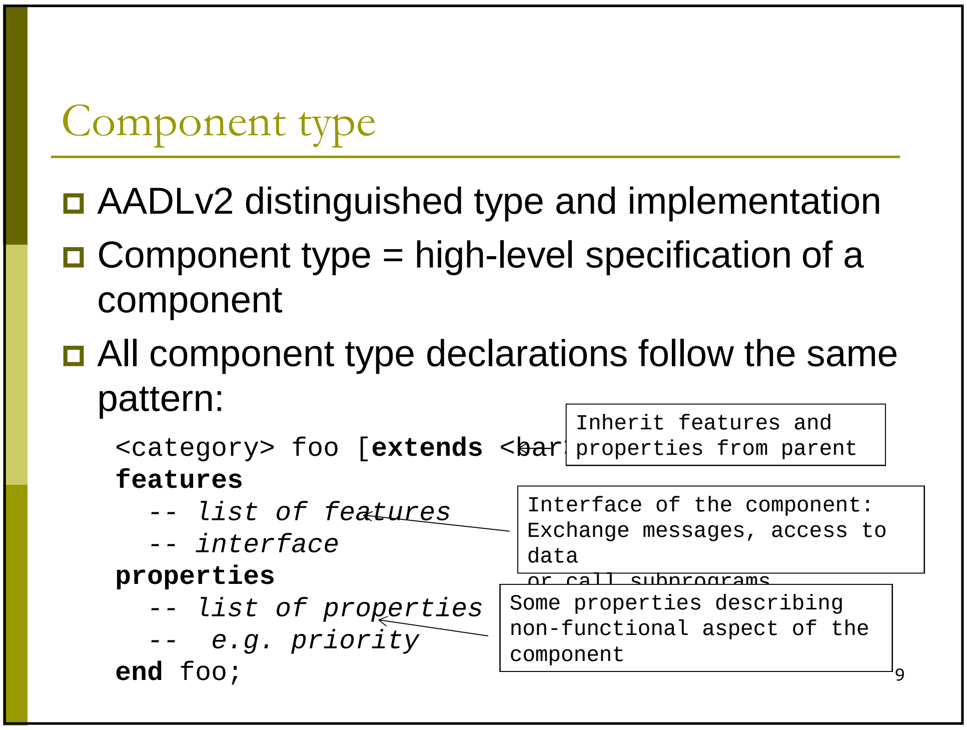
AADL components

- **AADL model** : hierarchy/tree of components
- **AADL component:**
  - Model a software or a hardware entity
  - May be organized in packages : **reusable**
  - Has a type/interface, one or several implementations
  - May have subcomponents
  - May combine/extend/refine others
  - May have properties : valued typed attributes (source code file name, priority, execution time, memory consumption, …)
- **Component interactions :**
  - Modeled by component connections
  - AADL features are connection points



AADL components

- **How to declare a component:**
  - Component type: name, category, properties, features => interface
  - Component implementation: internal structure (subcomponents), properties
- **Component categories:** model real-time abstractions, close to the implementation space (ex : processor, task, …). Each category has a well-defined semantics/behavior, refined through the property mechanism
  - Hardware components: execution platform
  - Software components
  - Systems : bounding box of a system. Model a deployment.



Component type

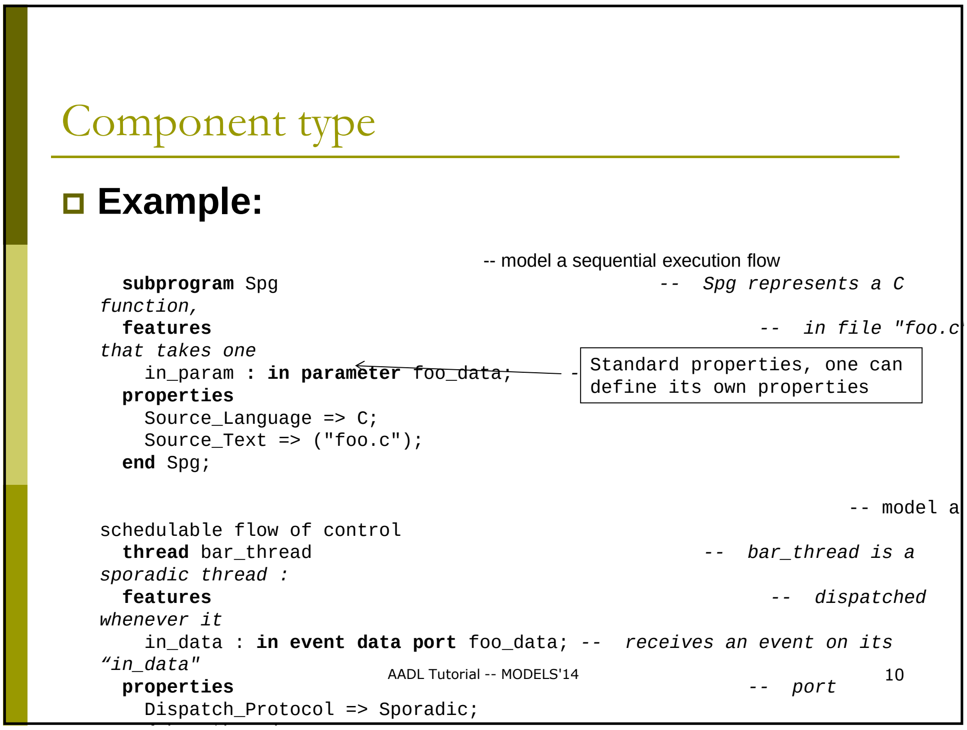
- AADLv2 distinguished type and implementation
- Component type = high-level specification of a component
- All component type declarations follow the same pattern:

```
<category> foo [extends <bar
features
  -- list of features
  -- interface
properties
  -- list of properties
  --  e.g. priority
end foo;
```

Inherit features and properties from parent

Interface of the component: Exchange messages, access to data or call subprograms

Some properties describing non-functional aspect of the component

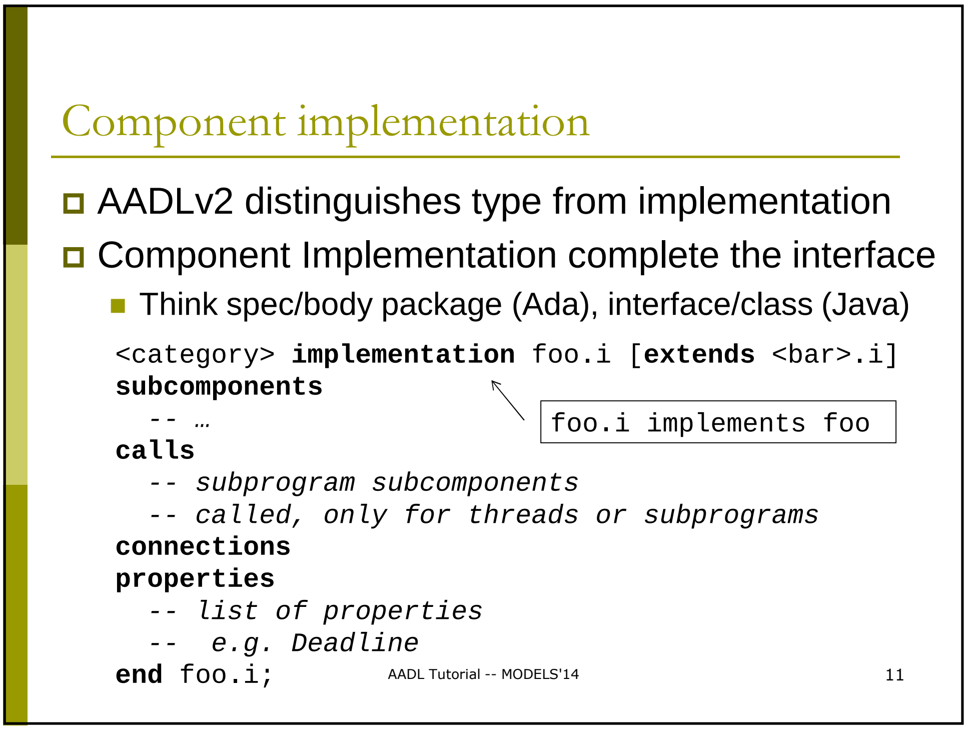


Component type

- **Example:**

```
                                        -- model a sequential execution flow
  subprogram Spg                                     --  Spg represents a C
function,
  features                                                --  in file "foo.c
that takes one
    in_param : in parameter foo_data;      -
  properties
    Source_Language => C;
    Source_Text => ("foo.c");
  end Spg;

                                                           -- model a
schedulable flow of control
  thread bar_thread                                   --  bar_thread is a
sporadic thread :
  features                                               --  dispatched
whenever it
    in_data : in event data port foo_data; --  receives an event on its
"in_data"
  properties                                            --  port
    Dispatch_Protocol => Sporadic;
```

Standard properties, one can define its own properties

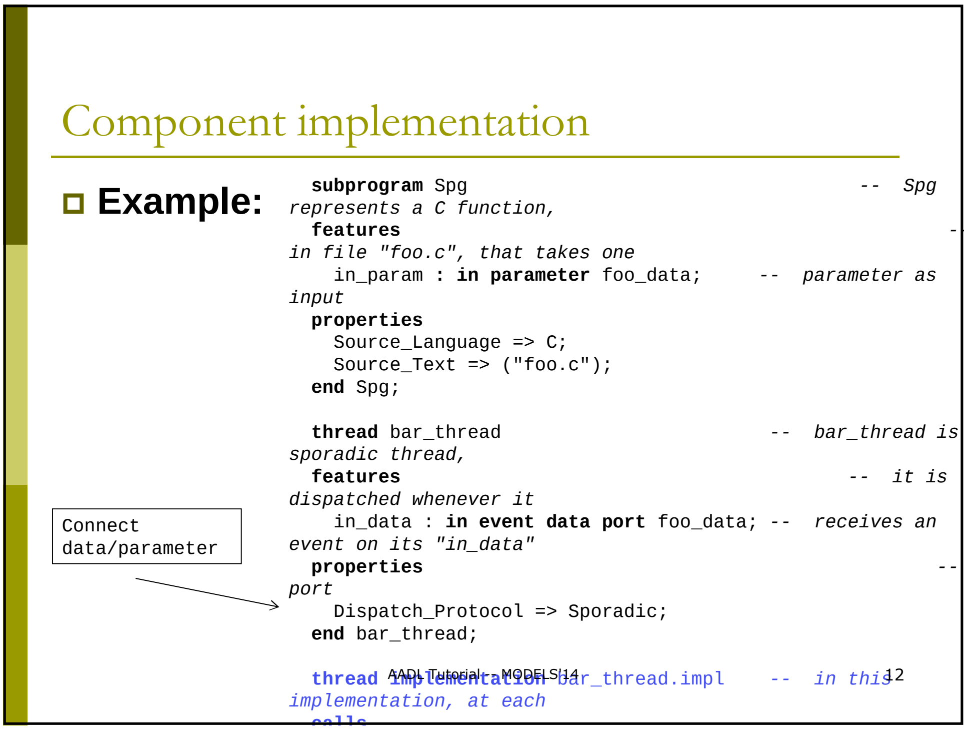


Component implementation

- AADLv2 distinguishes type from implementation
- Component Implementation complete the interface
  - Think spec/body package (Ada), interface/class (Java)

```
<category> implementation foo.i [extends <bar>.i]
subcomponents
  -- …
calls
  -- subprogram subcomponents
  -- called, only for threads or subprograms
connections
properties
  -- list of properties
  --  e.g. Deadline
end foo.i;
```

foo.i implements foo

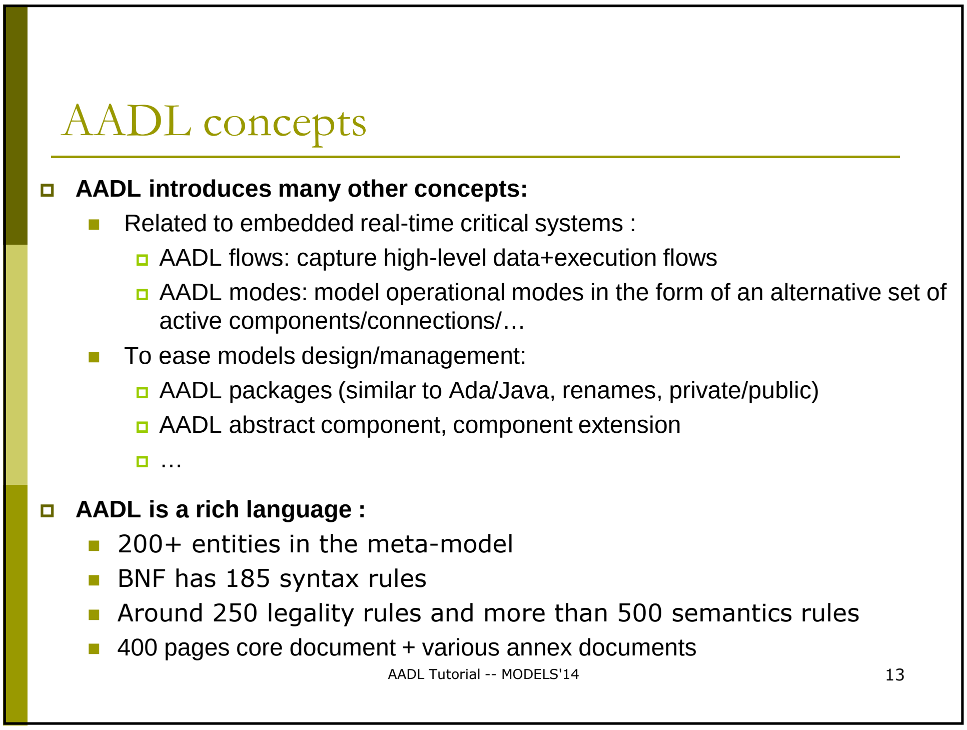


Component implementation

- **Example:**

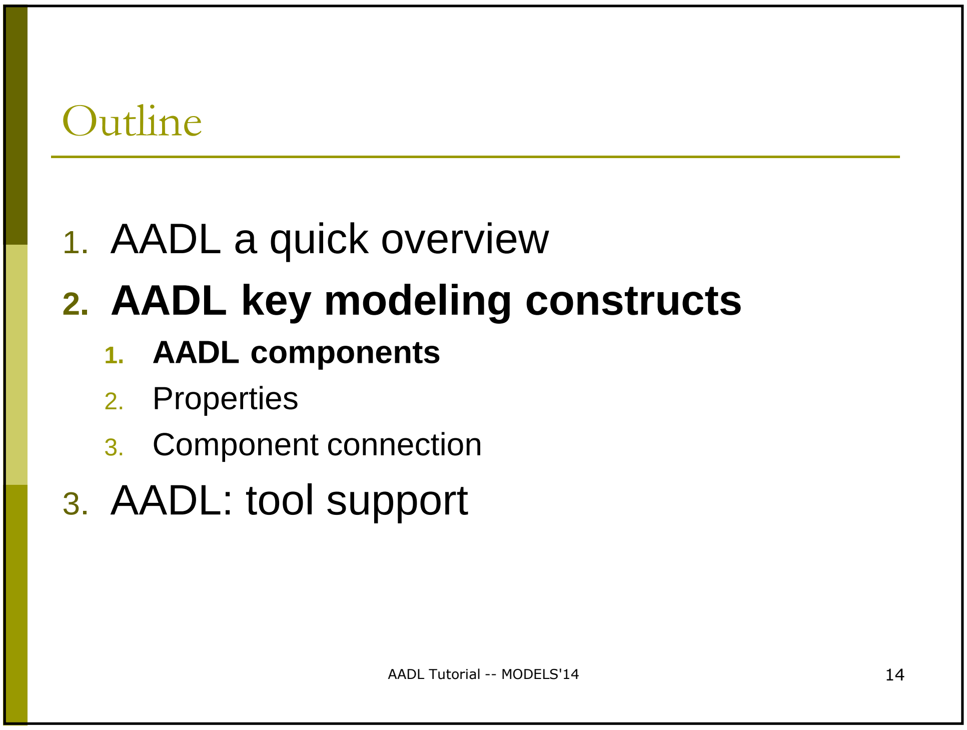
```
  subprogram Spg                                 --  Spg
represents a C function,
  features                                                --
in file "foo.c", that takes one
    in_param : in parameter foo_data;     --  parameter as
input
  properties
    Source_Language => C;
    Source_Text => ("foo.c");
  end Spg;

  thread bar_thread                          --  bar_thread is
sporadic thread,
  features                                      --  it is
dispatched whenever it
    in_data : in event data port foo_data; --  receives an
event on its "in_data"
  properties                                              --
port
    Dispatch_Protocol => Sporadic;
  end bar_thread;

  thread implementation bar_thread.impl    --  in this
implementation, at each
  calls
```

Connect data/parameter



AADL concepts

- **AADL introduces many other concepts:**
  - Related to embedded real-time critical systems :
    - AADL flows: capture high-level data+execution flows
    - AADL modes: model operational modes in the form of an alternative set of active components/connections/…
  - To ease models design/management:
    - AADL packages (similar to Ada/Java, renames, private/public)
    - AADL abstract component, component extension
    - …
- **AADL is a rich language :**
  - 200+ entities in the meta-model
  - BNF has 185 syntax rules
  - Around 250 legality rules and more than 500 semantics rules
  - 400 pages core document + various annex documents

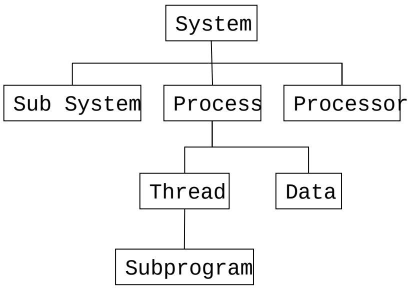


Outline

1. AADL a quick overview
2. **AADL key modeling constructs**
   1. **AADL components**
   2. Properties
   3. Component connection
3. AADL: tool support

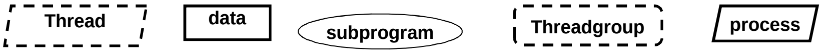
A full AADL system : a tree of component instances

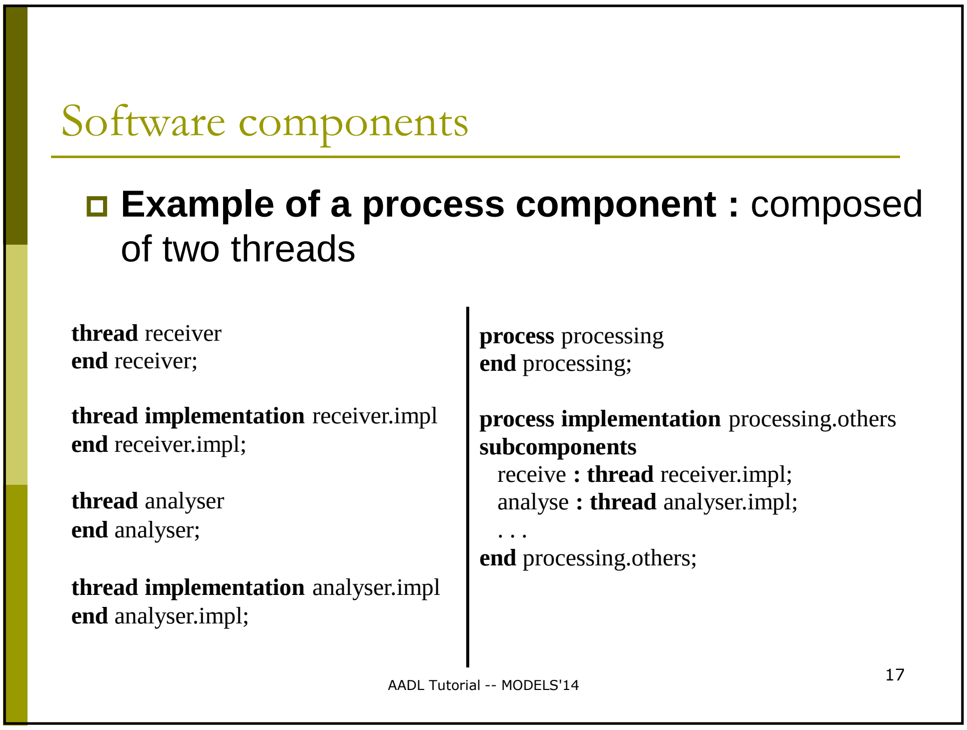
- Component types and implementations only define a library of entities (classifiers)
- An AADL model is a set of component instances (of the classifiers)
- System must be instantiated through a hierarchy of subcomponents, from root (system) to the leafs (subprograms, ..)
- We must choose a system implementation component as the root system model !



Software components categories

- **thread :** schedulable execution flow, Ada or VxWorks task, Java or POSIX thread. Execute programs
- **data :** data placeholder, e.g. C struct, C++ class, Ada record
- **process :** address space. It must hold at least one thread
- **subprogram :** a sequential execution flow. Associated to a source code (C, Ada) or a model (SCADE, Simulink)
- **thread group :** hierarchy of threads



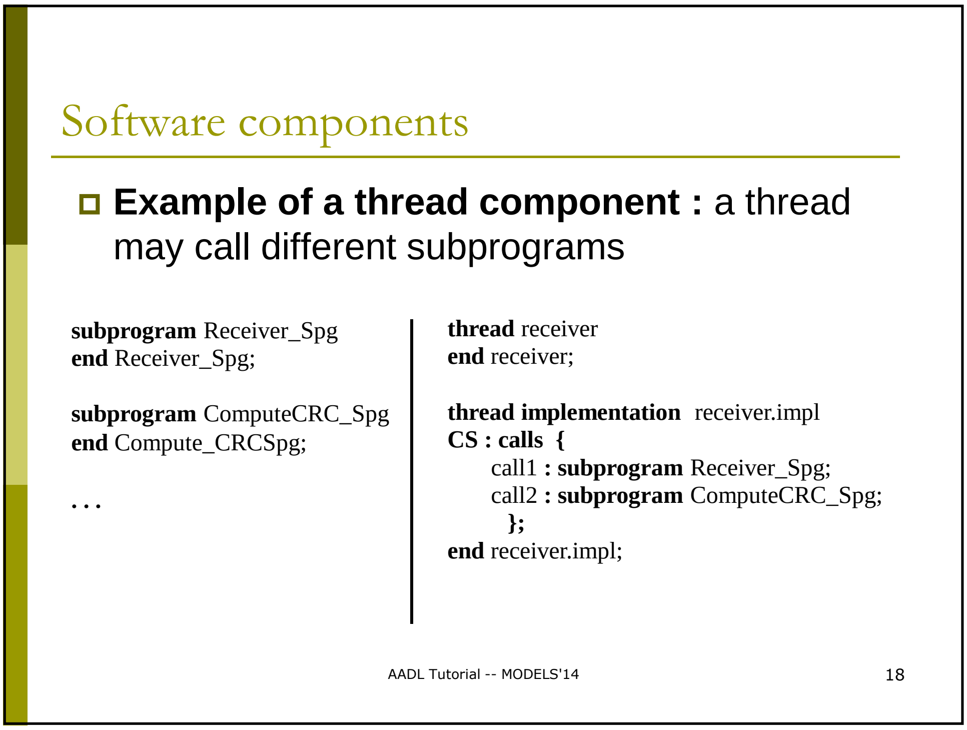


Software components

- **Example of a process component :** composed of two threads

```
thread receiver
end receiver;

thread implementation receiver.impl
end receiver.impl;

thread analyser
end analyser;

thread implementation analyser.impl
end analyser.impl;
```

```
process processing
end processing;

process implementation processing.others
subcomponents
   receive : thread receiver.impl;
   analyse : thread analyser.impl;
   . . .
end processing.others;
```

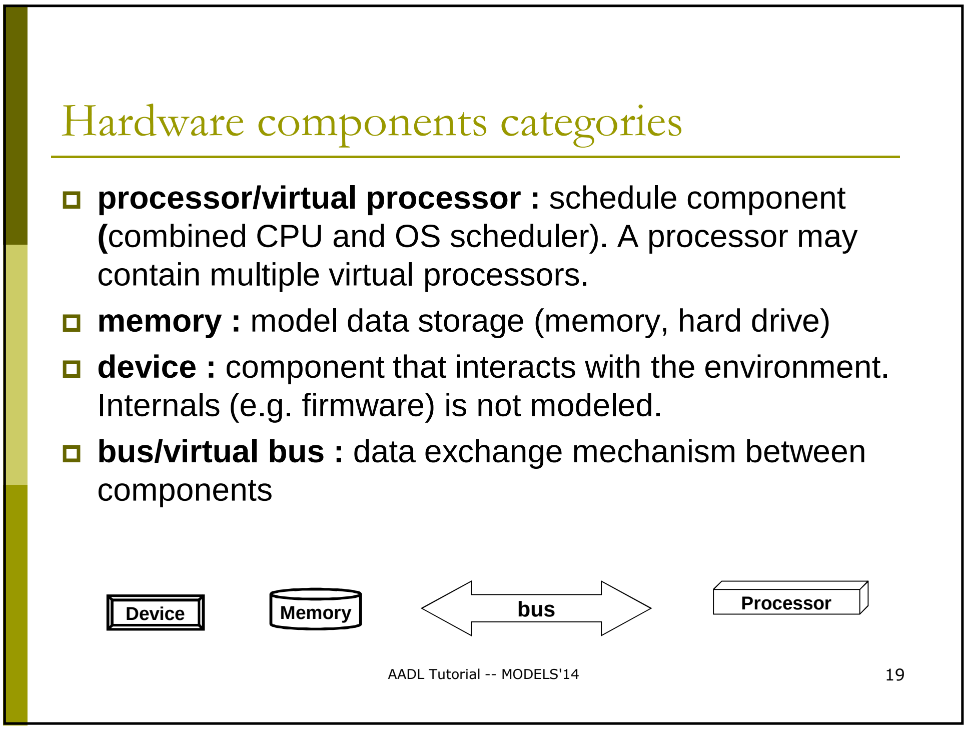


Software components

- **Example of a thread component :** a thread may call different subprograms

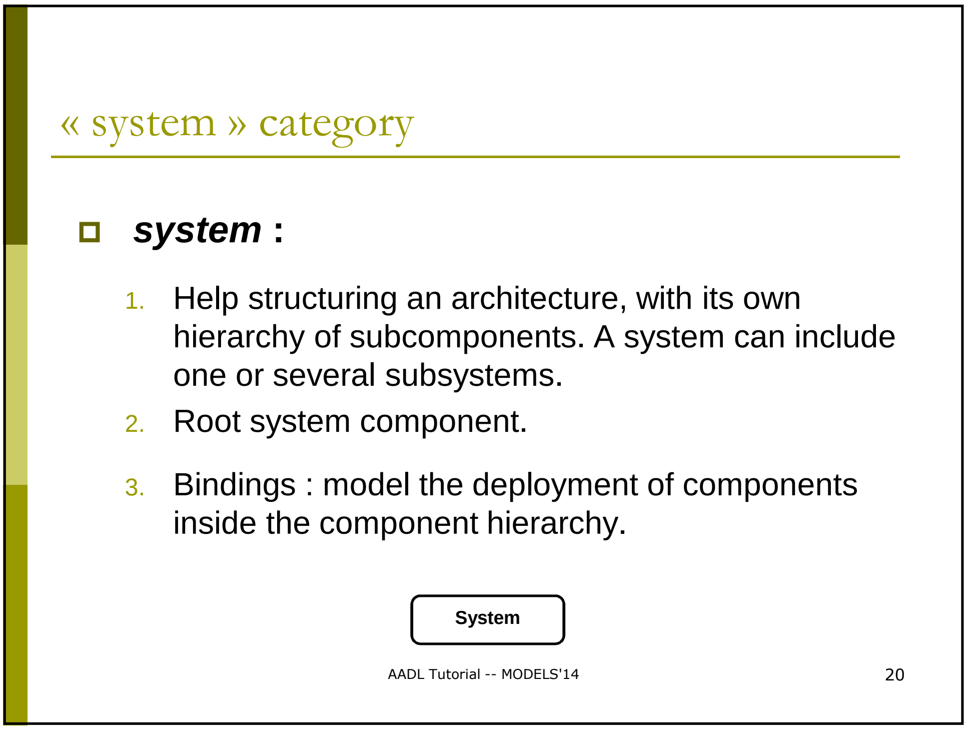
```
subprogram Receiver_Spg
end Receiver_Spg;

subprogram ComputeCRC_Spg
end Compute_CRCSpg;

. . .
```

```
thread receiver
end receiver;

thread implementation  receiver.impl
CS : calls  {
      call1 : subprogram Receiver_Spg;
      call2 : subprogram ComputeCRC_Spg;
        };
end receiver.impl;
```



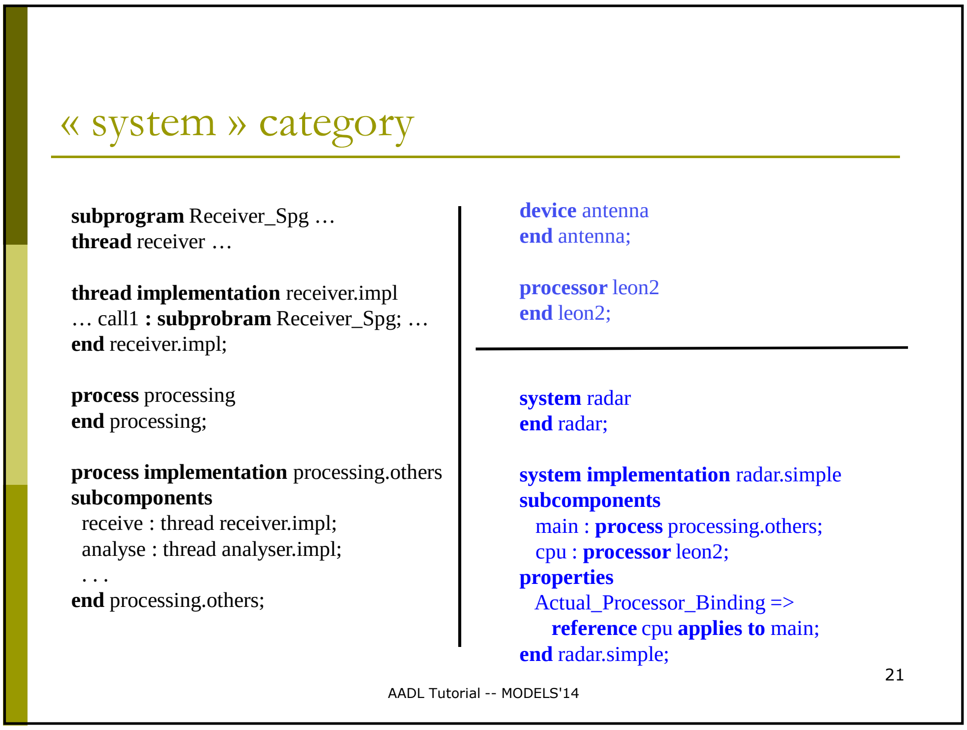
Hardware components categories

- **processor/virtual processor :** schedule component **(**combined CPU and OS scheduler). A processor may contain multiple virtual processors.
- **memory :** model data storage (memory, hard drive)
- **device :** component that interacts with the environment. Internals (e.g. firmware) is not modeled.
- **bus/virtual bus :** data exchange mechanism between components



« system » category

- ***system* :**
  1. Help structuring an architecture, with its own hierarchy of subcomponents. A system can include one or several subsystems.
  2. Root system component.
  3. Bindings : model the deployment of components inside the component hierarchy.

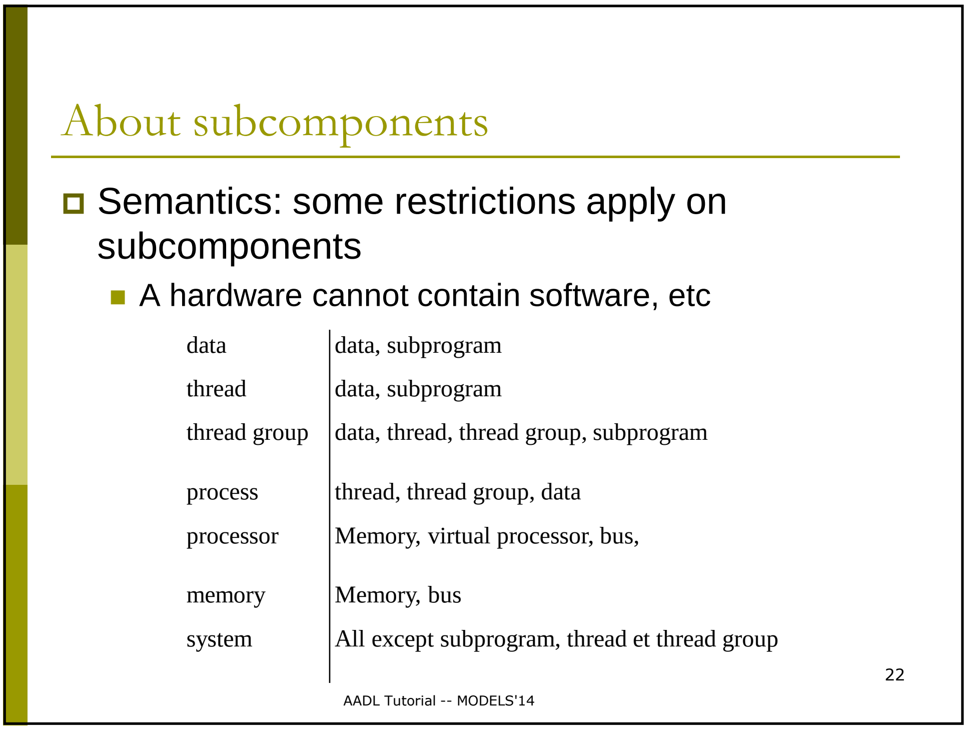


« system » category

```
subprogram Receiver_Spg …
thread receiver …

thread implementation receiver.impl
… call1 : subprobram Receiver_Spg; …
end receiver.impl;

process processing
end processing;

process implementation processing.others
subcomponents
  receive : thread receiver.impl;
  analyse : thread analyser.impl;
  . . .
end processing.others;
```

```
device antenna
end antenna;

processor leon2
end leon2;
```

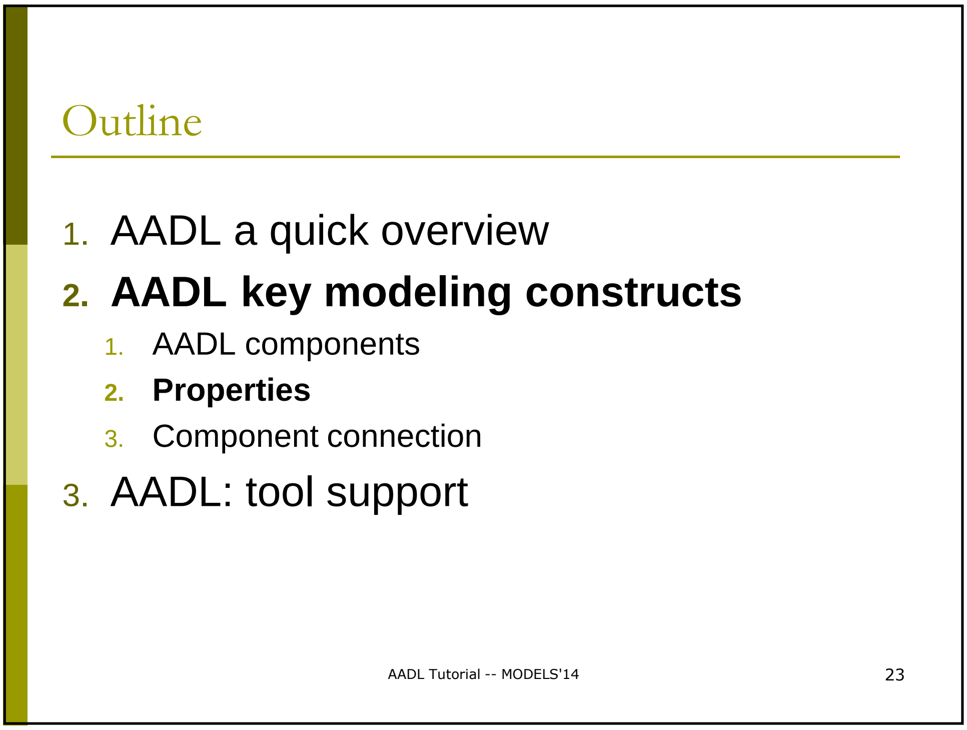
```
system radar
end radar;

system implementation radar.simple
subcomponents
   main : process processing.others;
   cpu : processor leon2;
properties
   Actual_Processor_Binding =>
      reference cpu applies to main;
end radar.simple;
```



About subcomponents

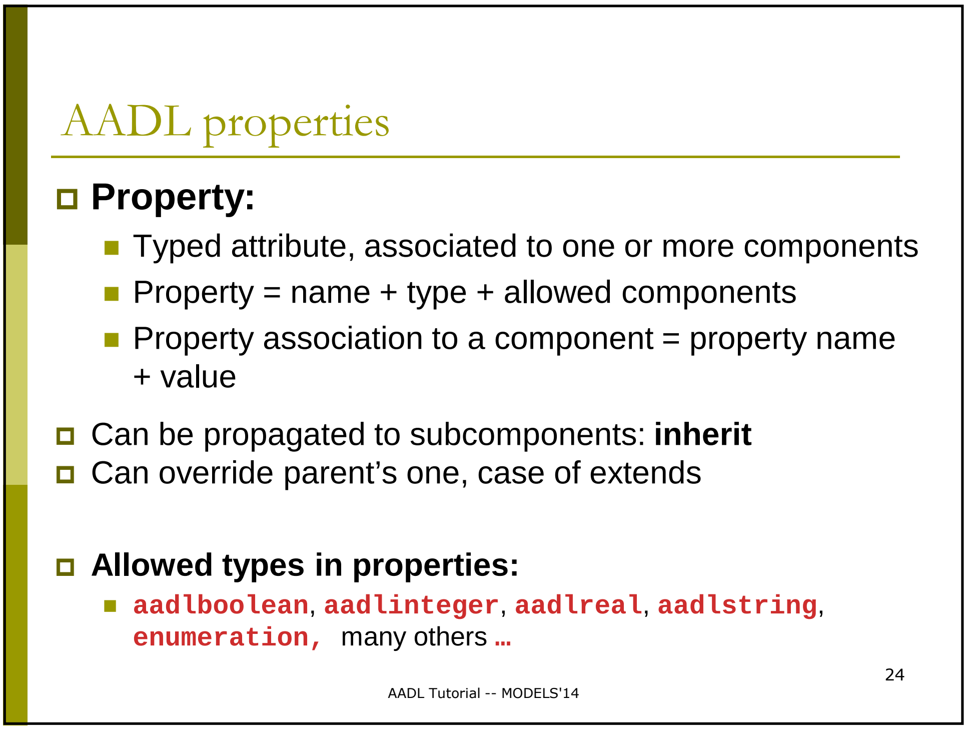
- Semantics: some restrictions apply on subcomponents
  - A hardware cannot contain software, etc

data	data, subprogram
thread	data, subprogram
thread group	data, thread, thread group, subprogram
process	thread, thread group, data
processor	Memory, virtual processor, bus,
memory	Memory, bus
system	All except subprogram, thread et thread group



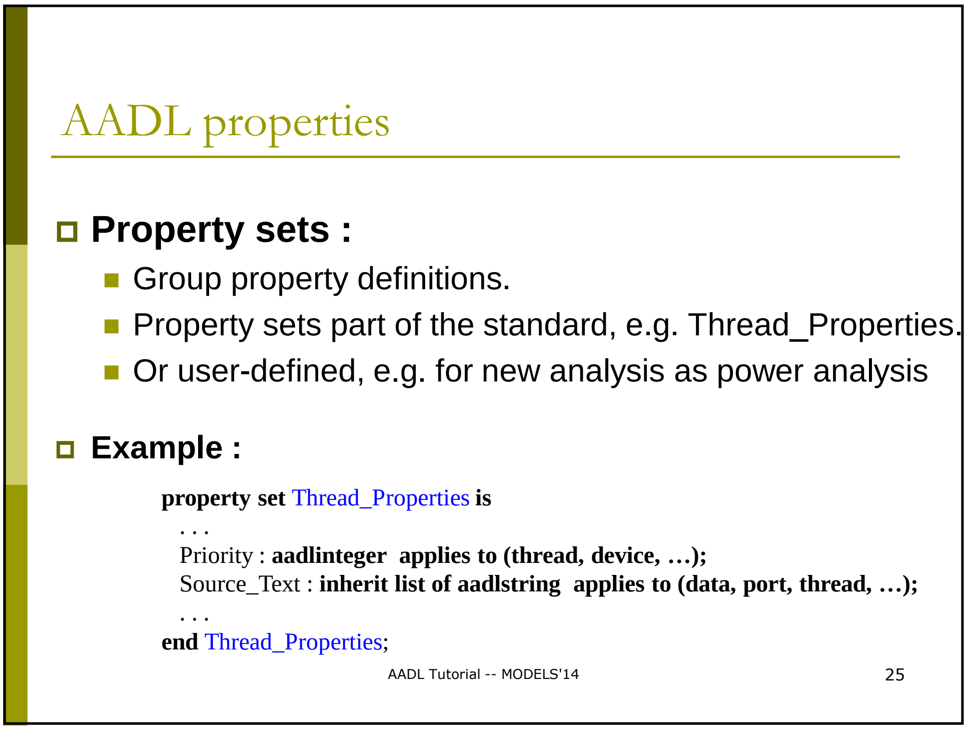
Outline

1. AADL a quick overview
2. **AADL key modeling constructs**
    1. AADL components
    2. **Properties**
    3. Component connection
3. AADL: tool support



AADL properties

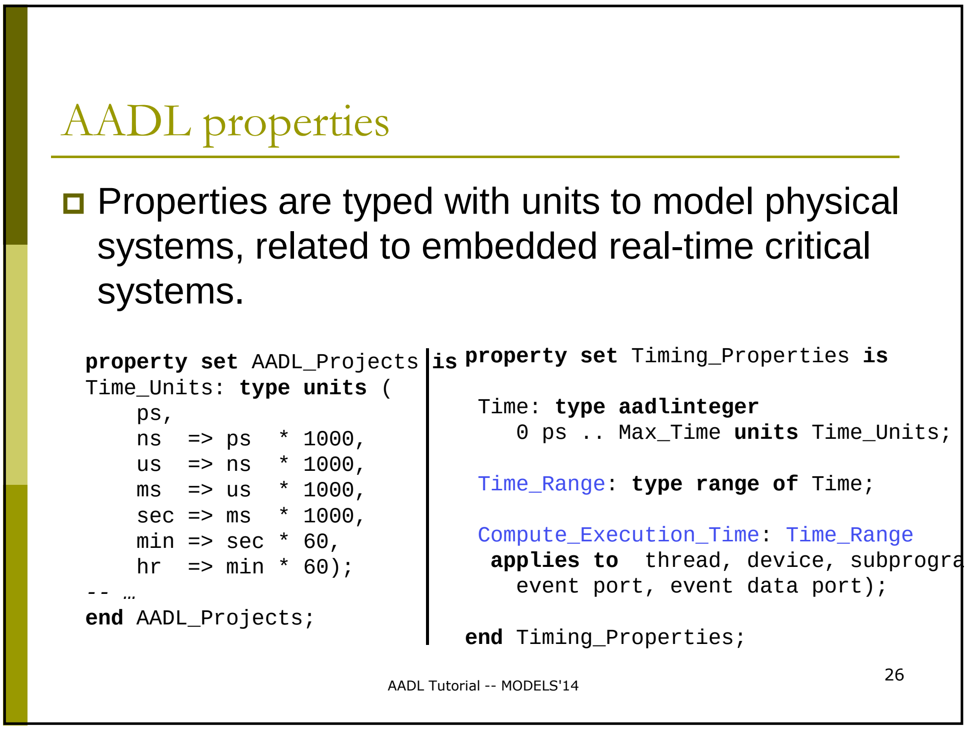
- **Property:**
    - Typed attribute, associated to one or more components
    - Property = name + type + allowed components
    - Property association to a component = property name + value
- Can be propagated to subcomponents: **inherit**
- Can override parent’s one, case of extends
- **Allowed types in properties:**
    - **`aadlboolean`**, **`aadlinteger`**, **`aadlreal`**, **`aadlstring`**, **`enumeration,`** many others **…**



AADL properties

- **Property sets :**
  - Group property definitions.
  - Property sets part of the standard, e.g. Thread_Properties.
  - Or user-defined, e.g. for new analysis as power analysis
- **Example :**

```
property set Thread_Properties is
  . . .
  Priority : aadlinteger  applies to (thread, device, …);
  Source_Text : inherit list of aadlstring  applies to (data, port, thread, …);
  . . .
end Thread_Properties;
```

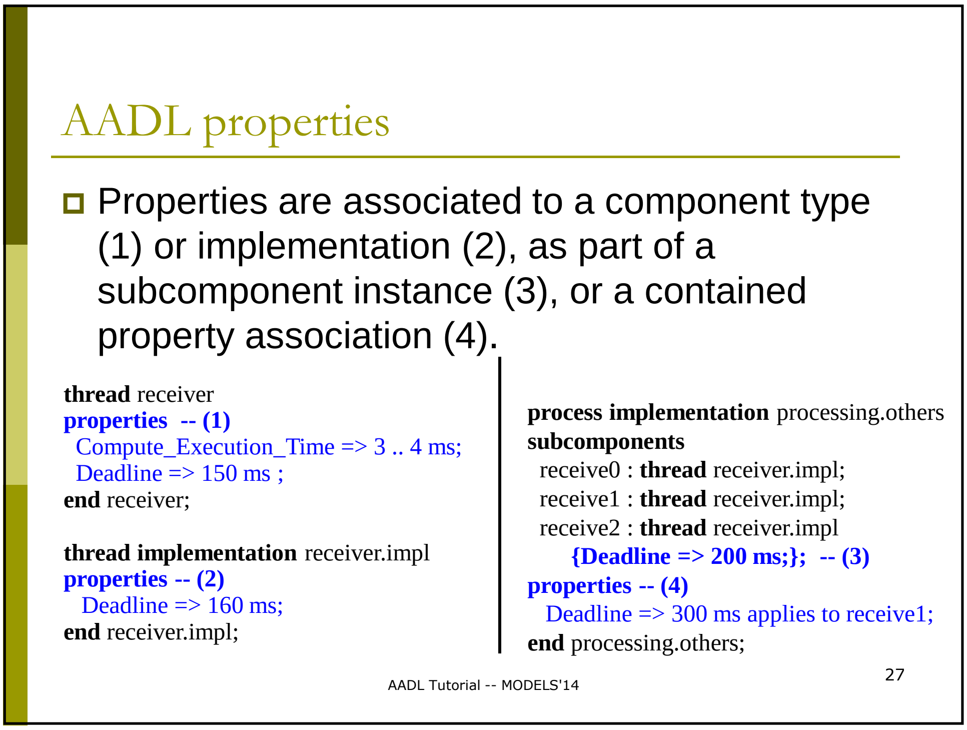


AADL properties

- Properties are typed with units to model physical systems, related to embedded real-time critical systems.

```
property set AADL_Projects is
Time_Units: type units (
    ps,
    ns  => ps  * 1000,
    us  => ns  * 1000,
    ms  => us  * 1000,
    sec => ms  * 1000,
    min => sec * 60,
    hr  => min * 60);
-- …
end AADL_Projects;
```

```
property set Timing_Properties is

 Time: type aadlinteger
    0 ps .. Max_Time units Time_Units;

 Time_Range: type range of Time;

 Compute_Execution_Time: Time_Range
  applies to  thread, device, subprogra
    event port, event data port);

end Timing_Properties;
```

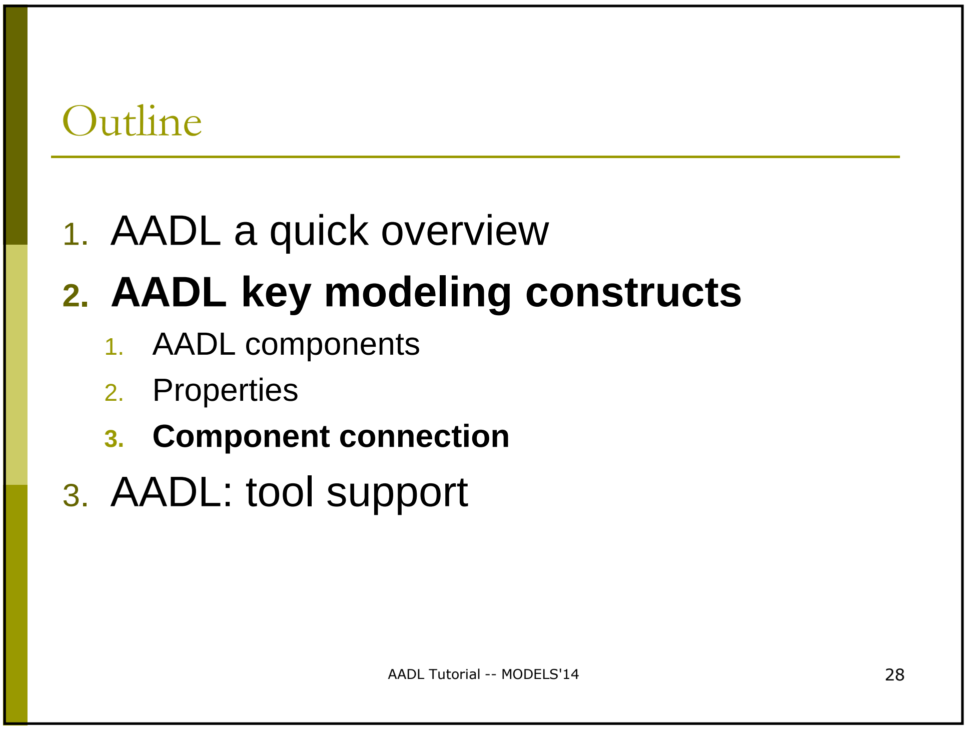


AADL properties

- Properties are associated to a component type (1) or implementation (2), as part of a subcomponent instance (3), or a contained property association (4).

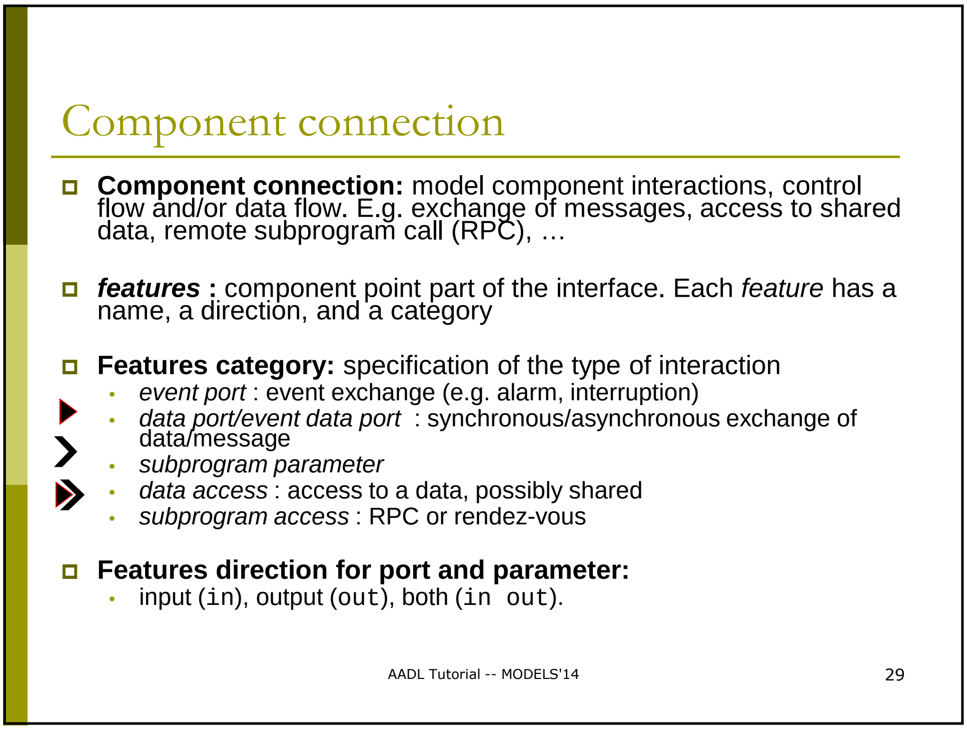
```
thread receiver
properties  -- (1)
  Compute_Execution_Time => 3 .. 4 ms;
  Deadline => 150 ms ;
end receiver;

thread implementation receiver.impl
properties -- (2)
   Deadline => 160 ms;
end receiver.impl;
```

```
process implementation processing.others
subcomponents
  receive0 : thread receiver.impl;
  receive1 : thread receiver.impl;
  receive2 : thread receiver.impl
       {Deadline => 200 ms;};  -- (3)
properties -- (4)
   Deadline => 300 ms applies to receive1;
end processing.others;
```



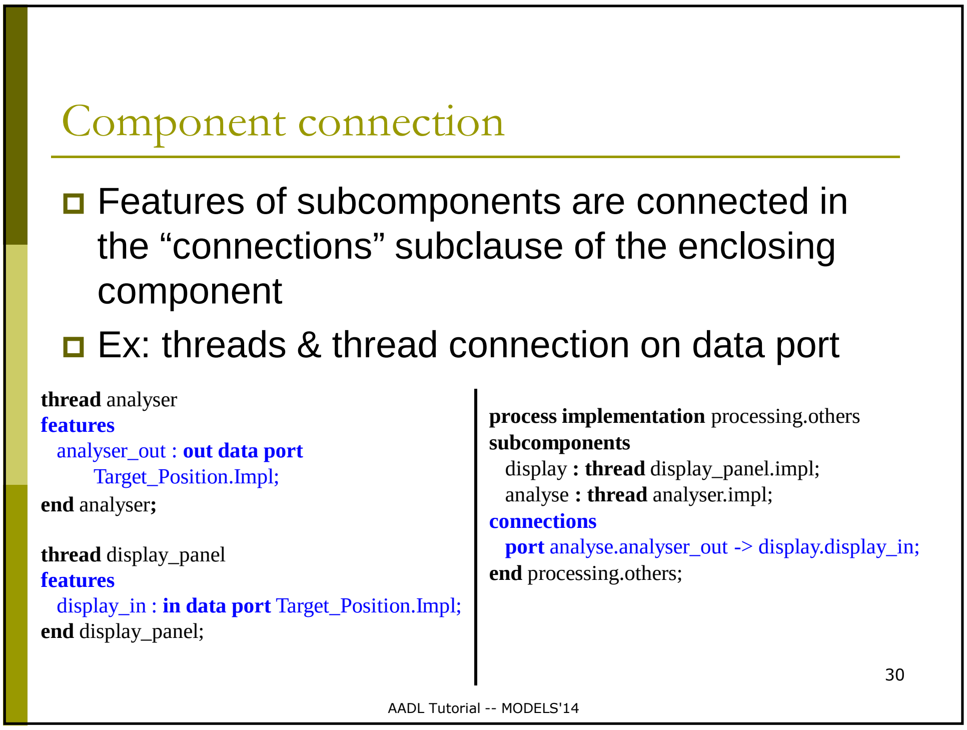
Outline

1. AADL a quick overview
2. **AADL key modeling constructs**
   1. AADL components
   2. Properties
   3. **Component connection**
3. AADL: tool support



Component connection

- **Component connection:** model component interactions, control flow and/or data flow. E.g. exchange of messages, access to shared data, remote subprogram call (RPC), …
- ***features* :** component point part of the interface. Each *feature* has a name, a direction, and a category
- **Features category:** specification of the type of interaction
  - *event port* : event exchange (e.g. alarm, interruption)
  - *data port/event data port* : synchronous/asynchronous exchange of data/message
  - *subprogram parameter*
  - *data access* : access to a data, possibly shared
  - *subprogram access* : RPC or rendez-vous
- **Features direction for port and parameter:**
  - input (`in`), output (`out`), both (`in out`).



Component connection

- Features of subcomponents are connected in the “connections” subclause of the enclosing component
- Ex: threads & thread connection on data port

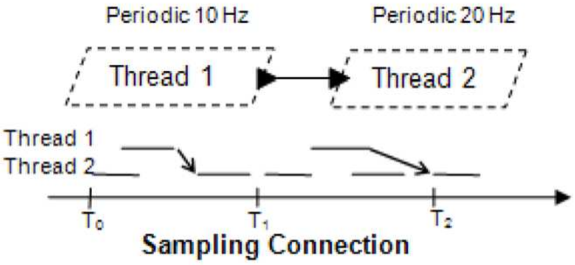
```
thread analyser
features
  analyser_out : out data port
       Target_Position.Impl;
end analyser;

thread display_panel
features
  display_in : in data port Target_Position.Impl;
end display_panel;
```

```
process implementation processing.others
subcomponents
  display : thread display_panel.impl;
  analyse : thread analyser.impl;
connections
  port analyse.analyser_out -> display.display_in;
end processing.others;
```

Data connection policies

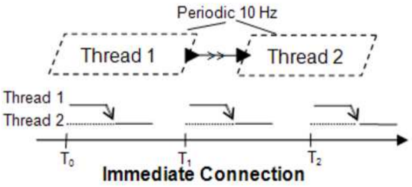
- **Allow deterministic communications**
- **Multiple policies exist to control production and consumption of data by threads:**

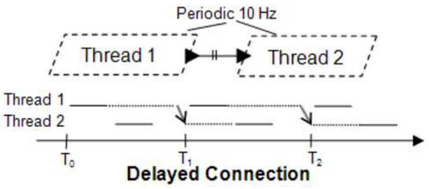
1. **Sampling connection:** takes the latest value
   - Problem: data consistency (lost or read twice) !

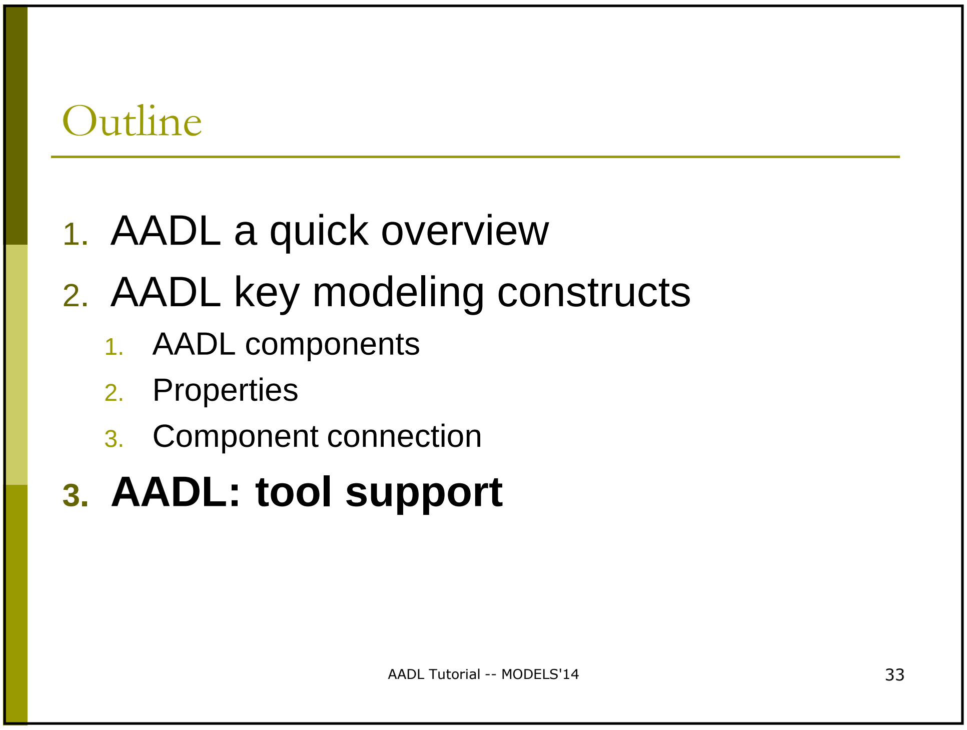


Data connection policies

2. **Immediate:** receiver thread is immediately awaken, and will read data when emitter finished
3. **Delayed:** actual transmission is delayed to the next time frame

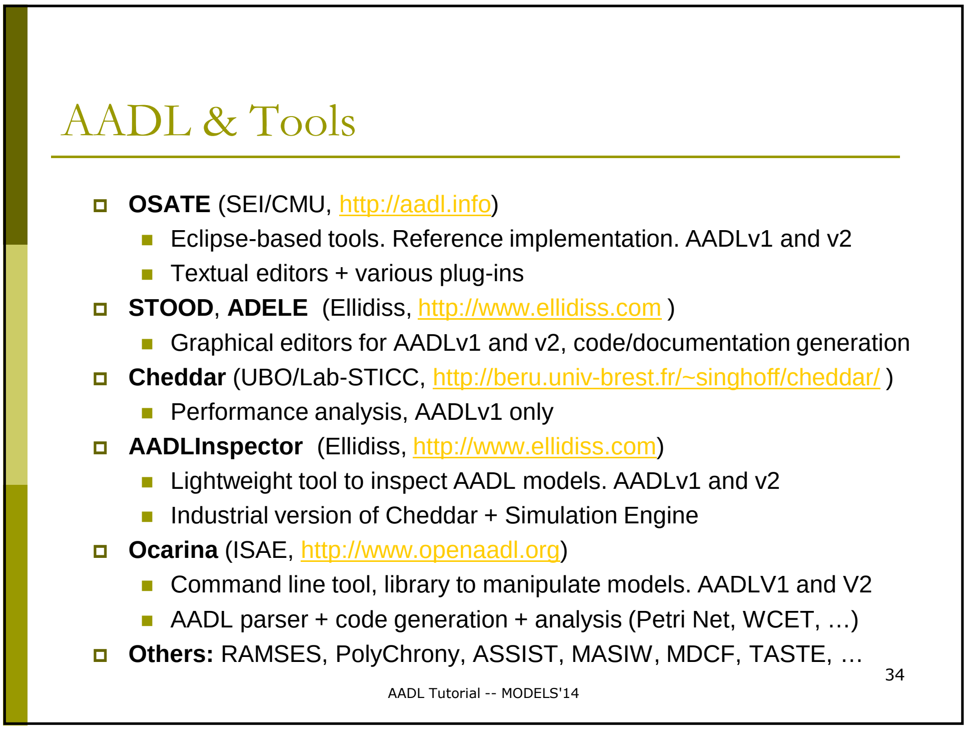






Outline

1. AADL a quick overview
2. AADL key modeling constructs
   1. AADL components
   2. Properties
   3. Component connection
3. **AADL: tool support**



AADL & Tools

- **OSATE** (SEI/CMU, http://aadl.info)
  - Eclipse-based tools. Reference implementation. AADLv1 and v2
  - Textual editors + various plug-ins
- **STOOD**, **ADELE** (Ellidiss, http://www.ellidiss.com)
  - Graphical editors for AADLv1 and v2, code/documentation generation
- **Cheddar** (UBO/Lab-STICC, http://beru.univ-brest.fr/~singhoff/cheddar/)
  - Performance analysis, AADLv1 only
- **AADLInspector** (Ellidiss, http://www.ellidiss.com)
  - Lightweight tool to inspect AADL models. AADLv1 and v2
  - Industrial version of Cheddar + Simulation Engine
- **Ocarina** (ISAE, http://www.openaadl.org)
  - Command line tool, library to manipulate models. AADLV1 and V2
  - AADL parser + code generation + analysis (Petri Net, WCET, …)
- **Others:** RAMSES, PolyChrony, ASSIST, MASIW, MDCF, TASTE, …