The Architecture Analysis and Design Language: an overview

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, deployments.
  - **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 the aerospace industry
- **Annex document to address specific needs**
  - Behavior, data, error modeling, code generation, …
- **AADL objectives are "to model a system"**
  - With analysis in mind
  - 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 components

- **AADL model : hierarchy/tree of components**
  - Textual, graphical representations, XMI serialization
- **AADL component models a software or a hardware entity**
  - May be organized in packages : **reusable**
  - Has a type/interface, one or several implementations
  - May have subcomponents
  - May extend/refine other components
  - 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

- **AADLv2 distinguished type and implementation**
  - Component type: high-level specification of a component
    - name, category, features, properties => interface
  - Component implementation: internal structure (subcomponents), additional or refined properties, connections
- **Component categories: model abstractions**
  - Categories have well-defined semantics, refined through properties
  - Denote software (threads, data, ..), hardware (processor, bus, ..)

Component type

> All component type declarations follow the same pattern:

```
<category> foo [extends <bar>]
features
  -- list of features, interface
  -- e.g. messages, access to data, etc.
properties
  -- list of properties, e.g. priority
end foo;
```

Inherit features and properties from parent

Some properties describing non-functional aspect of the component

```
 --  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;         --  parameter as input
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 port
properties
  Dispatch_Protocol => Sporadic;
end bar_thread;
```

Component implementation

> Component Implementation complete the interface

foo.i implements foo

```
<category> implementation foo.i [extends <bar>.i]
subcomponents
  -- internal elements
connections
  -- from external interface to internal subcomponents
properties
  -- list of properties
end foo.i;
```

```
  -- 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 port
  properties
    Dispatch_Protocol => Sporadic;
  end bar_thread;

  thread implementation bar_thread.impl      --  In this implementation, at each
  calls                                      --  dispatch we execute the "C" call
    C : { S : subprogram spg; };             --  sequence. We pass the dispatch
  connections                                --  parameter to the call sequence
    parameter in_data -> S.in_param;
  end bar_thread.impl;
```

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**

Software components categories

AADL component categories refer to well-known abstractions:

- **thread : schedulable entity, maps to task/thread of an RTOS**
- **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**

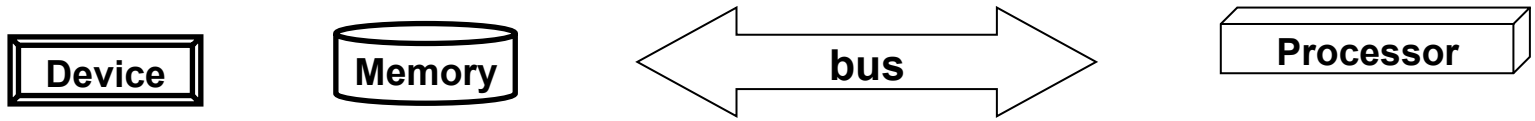
Component categories are attached to graphical elements:

Hardware components categories

Hardware categories model resources available:

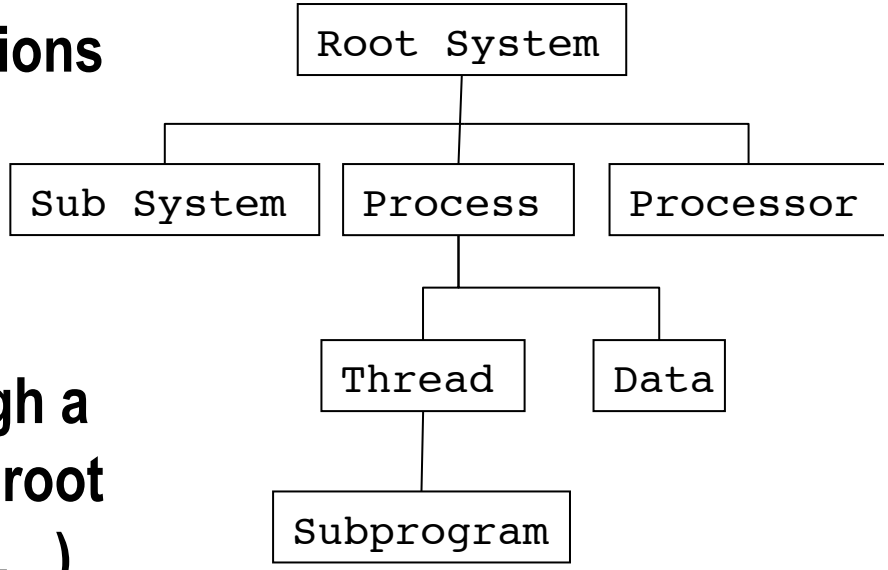
- processor/virtual processor : schedule component (combined CPU and RTOS scheduler). A processor may contain multiple virtual processors.
- memory : model data storage (memory, hard drive)
- device : component that interacts with the environment. Internals (e.g. firmware) is not modeled.
- bus/virtual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

Component categories are attached to graphical elements:



The system category

- Component types and implementations define a library of entities
- An AADL model is a set of component instances
- System must be instantiated through a hierarchy of subcomponents, from root (system) to the leafs (subprograms, ..) to define the actual system we analyze



- Captured using the “system” category

```
system ADIRU  end ADIRU;

system implementation ADIRU.impl
subcomponents
  main_mem  : memory ADIRU_memory::main_memory.impl;
  main_cpu  : processor ADIRU_processor::powerpc.impl;
```

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

  - Similar restrictions on semantic connections, binding of elements, etc.

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 components**
  - Property = name + type + allowed components
  - Property association = property name + value.
  - Can be propagated to subcomponents: **inherit**
  - Can override parent's one, case of extends
- **Property sets: group property definitions.**
  - Property sets part of the standard, e.g. Thread_Properties.
  - Or user-defined, e.g. for new analysis such as power analysis

```
property set Thread_Properties is
  Dispatch_Protocol: Supported_Dispatch_Protocols
    applies to (thread, device, virtual processor);

  Priority: inherit aadlinteger
    applies to (thread, thread group, process, system, device, data);
 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, subprogram
    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 foo
properties  -- (1)
  Compute_Execution_Time => 3 .. 4 ms;
  Deadline => 150 ms ;
end foo;

thread implementation foo.impl
properties -- (2)
   Deadline => 160 ms;
   Compute_Execution_Time => 4 .. 10 ms;
end foo.impl;
```

```
process implementation bar.others
subcomponents
  foo0 : thread foo.impl;
  foo1 : thread foo.impl;
  foo2 : thread foor.impl
      {Deadline => 200 ms;};  -- (3)
properties -- (4)
   Deadline => 300 ms applies to foo1;
end bar.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, shared data access, remote subprogram call (RPC), …
- ***features* : component point part of the interface**
  - Each *feature* has a name, a direction, and a category
- **Features direction for port and parameter:**
  - input (in), output (out), both (in out)
- **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

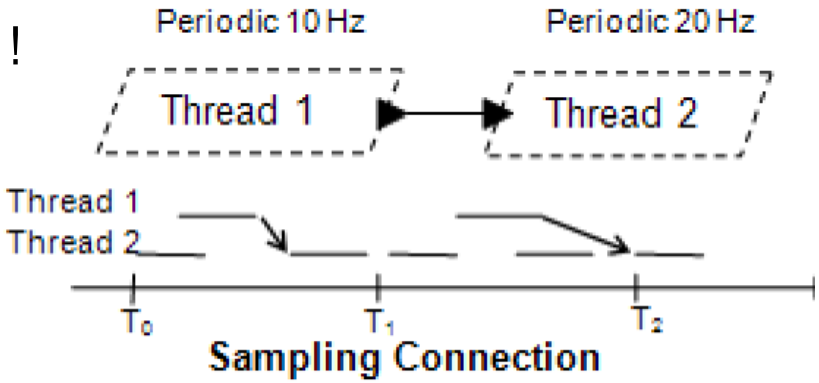
Component connection

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

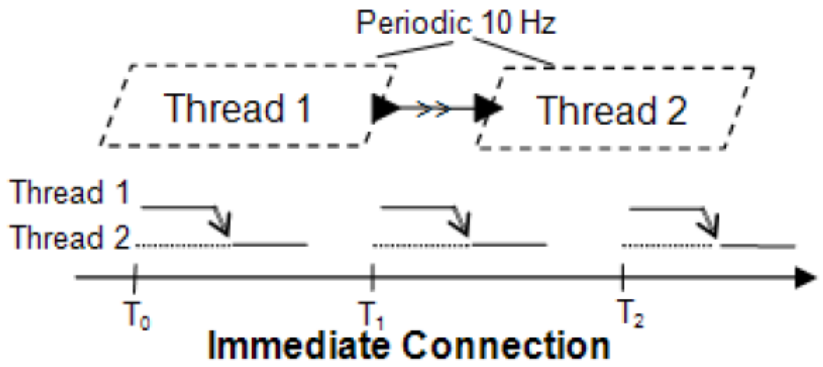
```
process acc_process
features
  acc1_output: out data port SHM_DataType::accData;
  -- …
end acc_process;

process implementation acc_process.impl
subcomponents
  acc1: thread threads::acc1_dataOutput.impl;
  -- …
connections
  C7: port acc1.acc1out -> acc1_output;
  -- ..
```

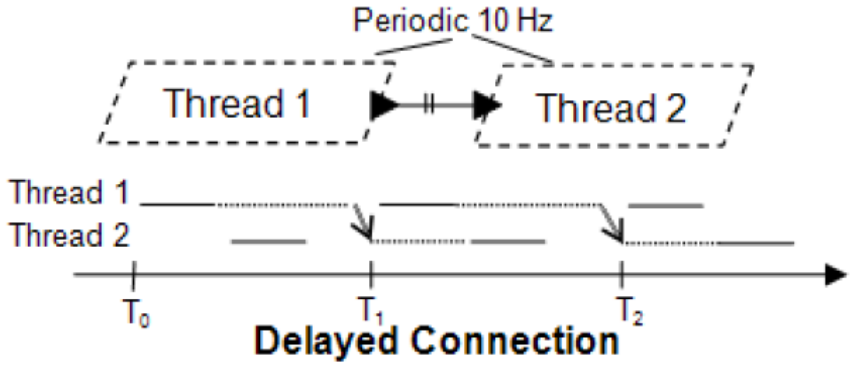
Data connection policies

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



2. **Immediate: receiver thread is immediately awaken, and will read data when emitter finishes**
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 toolchains

- **Multiple AADL toolchains exist, they can be easily combined via the textual syntax. Most of them have a FLOSS license**
- **OSATE (SEI/CMU, http://aadl.info)**
  - Eclipse-based tools. Reference implementation
  - Textual and graphical editors + various plug-ins (latency, security, …)
- **Ocarina (ISAE, http://www.openaadl.org)**
  - Command line tool, library to manipulate models in Python
  - AADL parser + code generation + analysis (Petri Net, WCET, …)
- **AADLInspector (Ellidiss, http://www.ellidiss.com)**
  - Lightweight tool to inspect AADL models. AADLv1 and v2
  - Industrial version of Cheddar + Simulation Engine
- **Others: RAMSES, PolyChrony, ASSIST, MASIW, MDCF, TASTE, …**

- **In the following, we will concentrate on OSATE and Ocarina**