a partitioned pok operating system

POK – System specification

Julien Delange <julien@gunnm.org>

Forewords

a partitioned pok operating system

- **The POK project**
  - **Design and implement safe and secure system**
  - **Complete development process with model-based engineering**
- **Now, focus on system specification**
  - **How you can describe system requirements**
  - **Different representation approaches**
  - **Validation functionalities, requirements enforcement**

Outline

a partitioned pok operating system

- Reminder about partitioned architectures
- Introduction to XML and AADL formalisms
- AADL specification and validation
- XML specification and validation

Outline

a partitioned pok operating system

- **Reminder about partitioned architectures**
- **Introduction to XML and AADL formalisms**
- **AADL specification and validation**
- **XML specification and validation**

Reminder

a partitioned pok operating system

- **Strong requirements**
  - **Partitions run as if they are on a single processor**
- **Time isolation**
  - **Partitions has a fixed time slice**
  - **Two scheduling layers**
- **Space isolation**
  - **Partitions contained in isolated address spaces**
  - **Communications monitored by the kernel**

Specification rationale

a partitioned pok operating system

- **Describe system requirements**
  - **Kernel level : isolation policy, required functionalities, …**
  - **Partition level : resources, communications channels, …**
- **Analyze**
  - **Detect errors and defects as early as possible**
  - **Avoid errors prior to implementation steps**
- **And more !**
  - **Generate configuration or code**
  - **Use them for certification purposes**

Specification requirements

a partitioned pok operating system

- **Appropriate semantics**
  - **Declare architecture/requirements as more as possible**
  - **Specify kernel and partitions requirements …**
  - **… and potentially other nodes**
- **Avoid disambiguation**
  - **One requirement = one specification pattern**
  - **Help tools in specifications usage**

Outline

a partitioned pok operating system

- Reminder about partitioned architectures
- **Introduction to XML and AADL formalisms**
- AADL specification and validation
- XML specification and validation

Formalisms

a partitioned pok operating system

- **AADL**
  - **Architecture modeling standard, standardized by the SAE**
  - **Dedicated patterns for partitioned architectures**
  - **Appropriate semantics for system specification**
- **XML**
  - **ARINC653 specific file content**
  - **Wide-known formalism, many tools**
  - **General-purpose semantics, not specific to modeling**

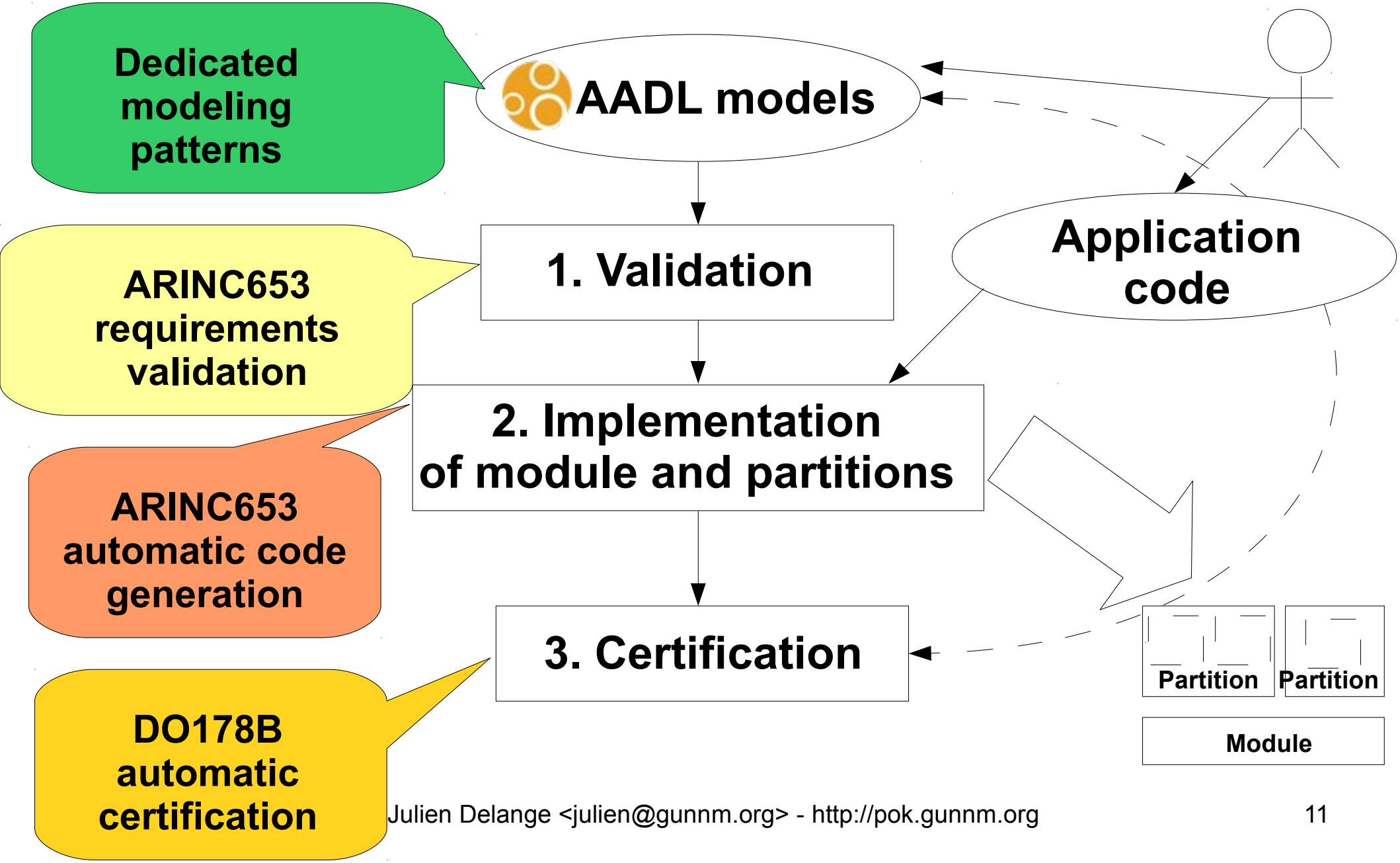
AADL vs. XML

a partitioned pok operating system

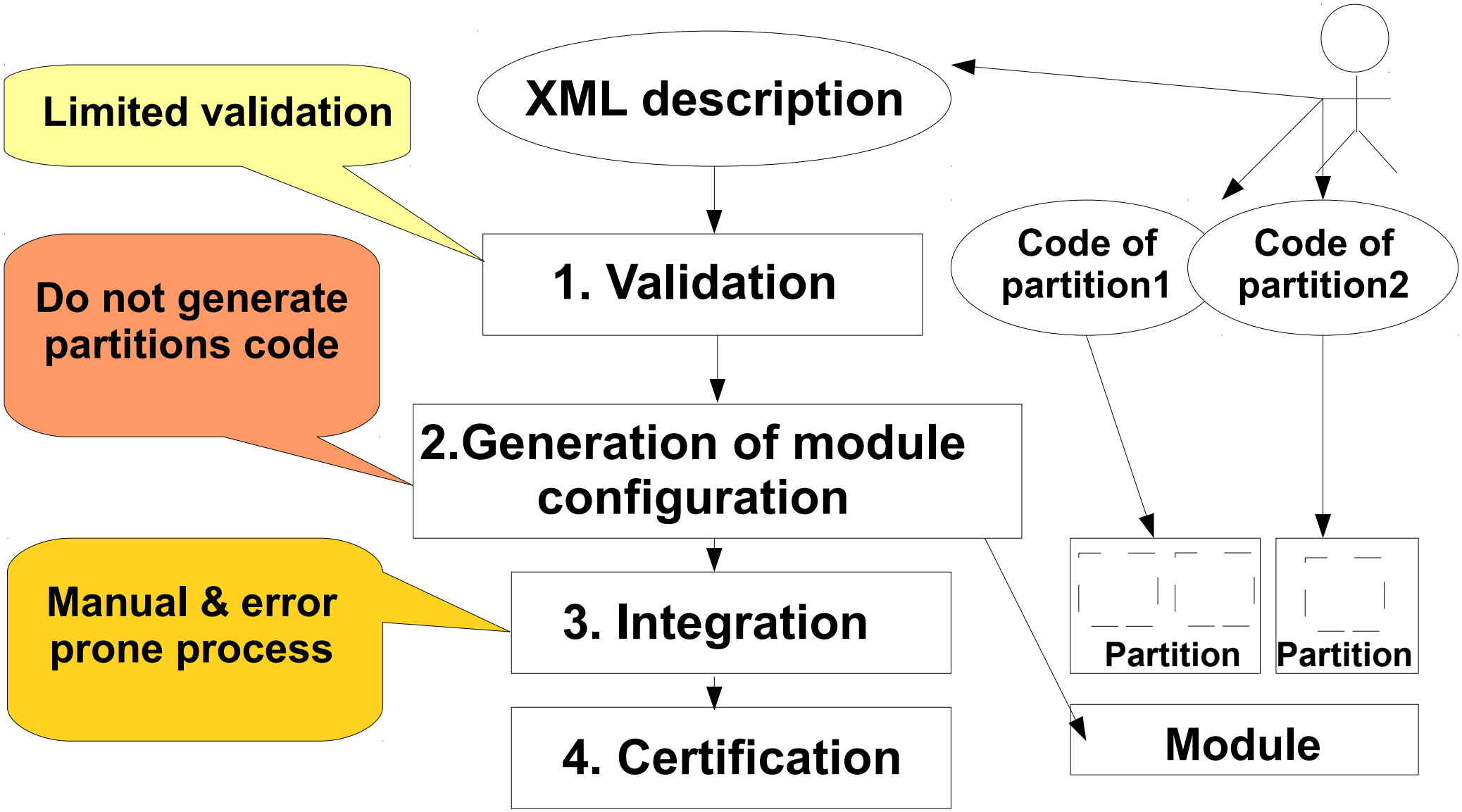
	Pros	Cons
AADL	• Complete system specs. • Graphical and textual representation • System validation tools • Code Generation for configuration and behavioral code • Certification hints	• Still “new” technology • O/S specific code generation
XML	• Many editors • Supported by all ARINC653 implementations • Generate kernel configuration	• Textual representation only • Do not describe partitions requirements • Do not generate behavioral code • Partial system description

AADL development process

a partitioned pok operating system



XML development process



Tool support

a partitioned pok operating system

- **AADL**
  - **Modeling: Eclipse/OSATE, Topcased, Ocarina, …**
  - **System analysis: OSATE, Ocarina/REAL**
  - **Code Generation: Ocarina**
- **XML**
  - **Modeling : text-editor, no graphical representation**
  - **System analysis: O/S specific**
  - **Code Generation: O/S specific**

Outline

a partitioned pok operating system

- Reminder about partitioned architectures
- Introduction to XML and AADL formalisms
- **AADL specification and validation**
- XML specification and validation

AADL basics

a partitioned pok operating system

- **Convenient modeling approaches**
  - **Graphical notation : user-friendly**
  - **Textual notation : tool-oriented**
- **Aggregation of components**
  - **Topmost component : a system**
  - **System contains other components**

AADL language basics

a partitioned pok operating system

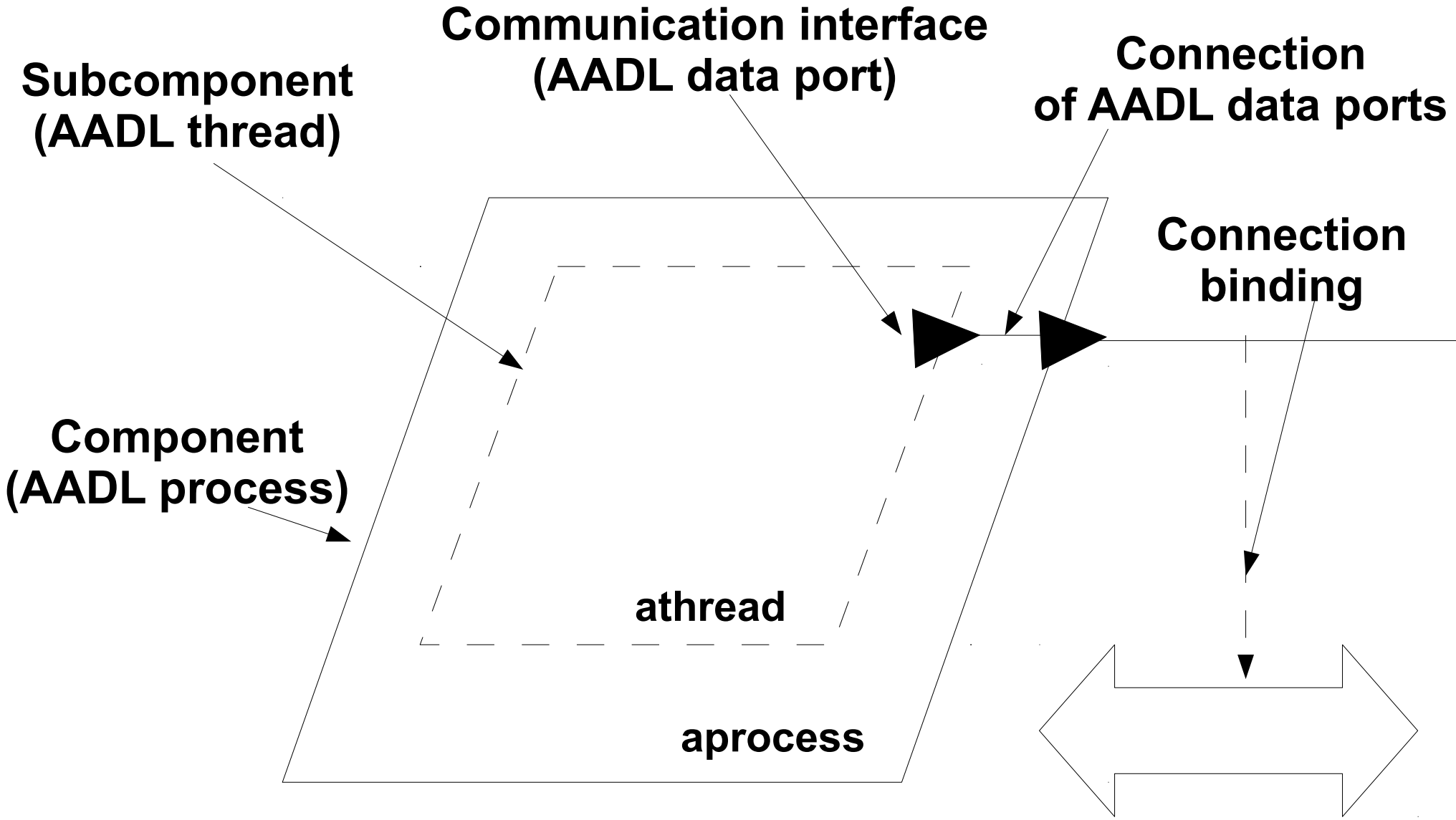
- **Components categories**
  - **Software : subprogram, thread, process, data**
  - **Hardware : processor, memory, bus, device, virtual processor, virtual bus**
  - **Other : abstract, system**
- **Extensions mechanisms**
  - **Properties bound to each component**
  - **Annex languages (behavior, error, ...)**

AADL language basics

a partitioned pok operating system

- Components and subcomponents
  - Components contains other components
  - (ex : a process contains one or several threads)
  - Constrained by predefined standardized rules
- Communication mechanisms (interfaces)
  - Components export communication ports
  - Ports are potentially associated with a type
  - Component requires/provides access to something
  - Ex : A process need an access to a bus

AADL model example



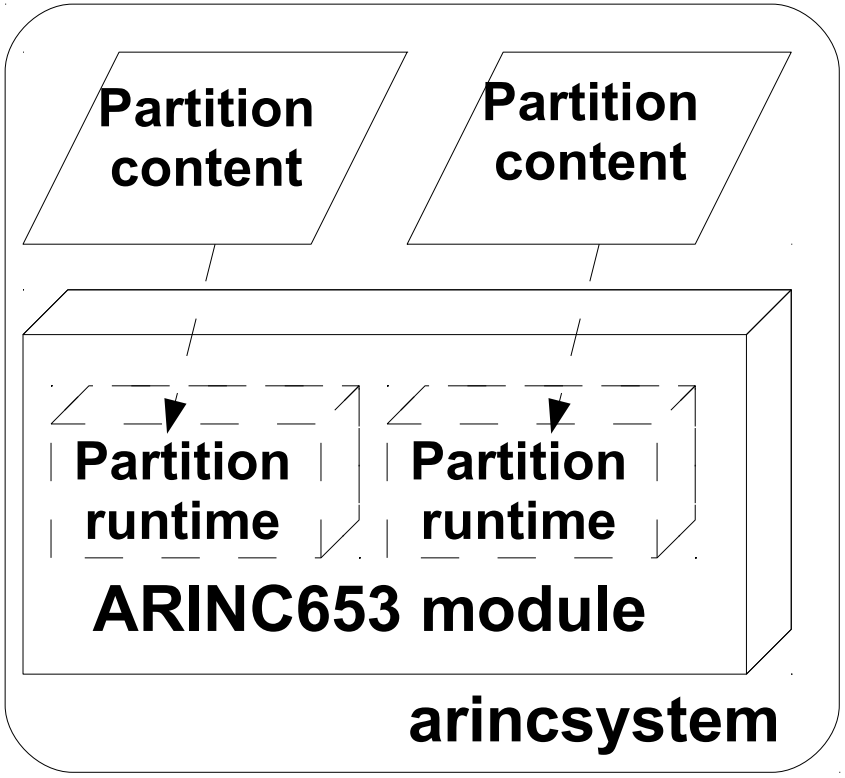
ARINC653 specification

a partitioned pok operating system

- **Dedicated annex of the AADL**
  - **Standardized modeling patterns**
  - **Experienced for configuration and code generation**
- **Introduce specific properties**
  - **ARINC653 requirements**
  - **Helpful for ARINC653 systems generation**

Partition modeling

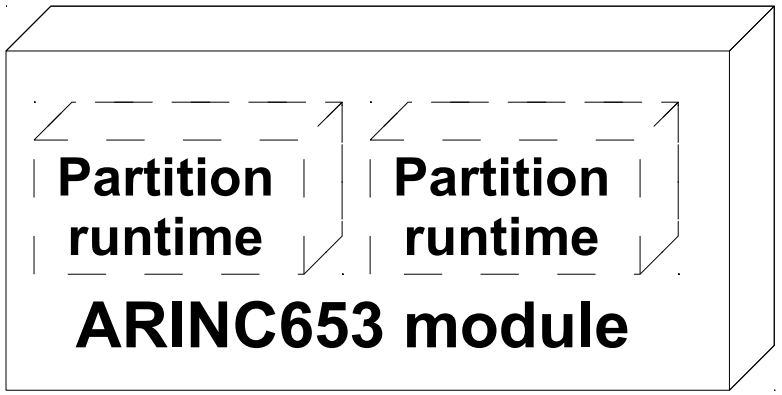
- **AADL Process + virtual processor components**
  - **Process = partition content (processes, ...)**
  - **Virtual processor = partition runtime (scheduler, services, ...)**
  - **Association with the `Actual_Processor_Binding` property**




```
system implementation arincsystem.i
subcomponents
  module : processor arincmodule.i;
  partcontent1 : process part1.i;
  partcontent2 : process part2.i;
properties
  Actual_Processor_Binding =>
    (reference (module.part1))
     applies to partcontent1;
  Actual_Processor_Binding =>
    (reference (module.part2))
     applies to partcontent2.i;
end arincsystem.i;
```

Module modeling

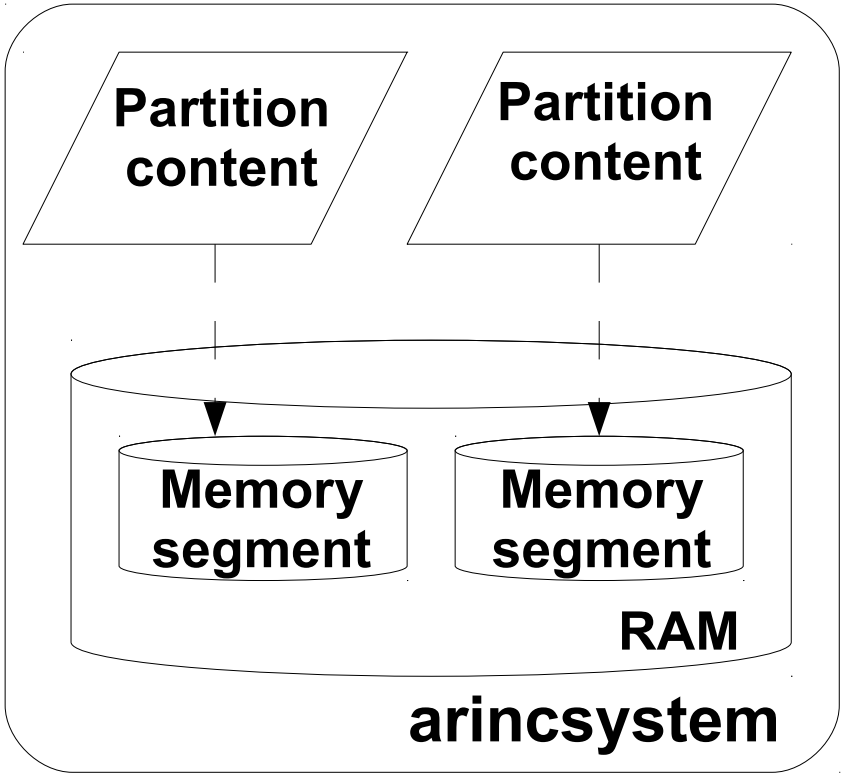
- **AADL Processor component**
  - **Contains partitions runtime (virtual processor)**
  - **Time partitioning policy using specific properties**




```
processor implementation module.i
subcomponents
  part1 : virtual processor
          partruntime.i;
  part2 : virtual processor
          partruntime.i;
properties
  ARINC653::Partitions_Slots =>
    (300ms, 500ms);
  ARINC653::Slots_Allocation =>
   (reference (part1),
    reference (part2));
end module.i;
```

Memory modeling/space isolation

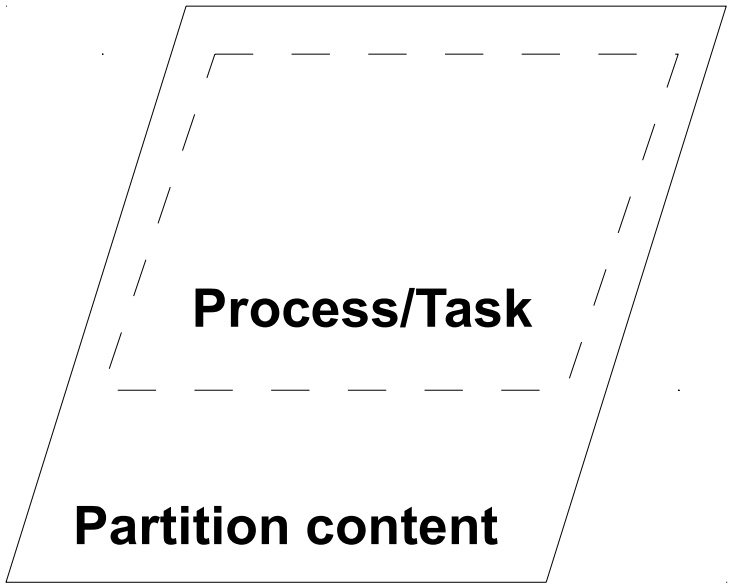
- **Hierarchy of AADL memory components**
  - **Main memory component = RAM**
  - **Memory subcomponents = memory segments**
  - **Associate partitions with memory segments with the `Actual_Memory_Binding` property**




```
system implementation arincsystem.i
subcomponents
  mainmem : memory ram.i;
  partcontent1 : process part1.i;
  partcontent2 : process part2.i;
properties
  Actual_Memory_Binding =>
    (reference (mainmem.seg1))
     applies to partcontent1;
  Actual_Memory_Binding =>
    (reference (mainmem.seg2))
     applies to partcontent2;
end arincsystem.i;
```

Process modeling

- **AADL Thread in process components**
  - **Model instruction flows**
  - **Thread constraints specified using standard properties**




```
thread implementation arincprocess.i
properties
  Period => 10ms;
  Compute_Execution_Time => 0ms .. 10ms;
  Deadline => 10ms;
end arincprocess.i;

process implementation partitioncontent.i
subcomponents
  myprocess : thread arincprocess.i;
end partitioncontent.i;
```

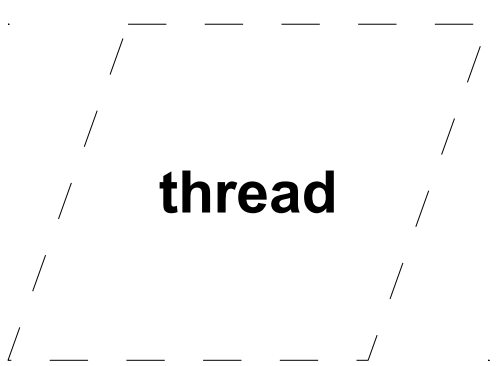
Interface with application code

a partitioned pok operating system

- **Subprogram calls inside a thread**
  - **AADL Subprogram components reference application code**
  - **AADL Thread components describe their call sequence**

```
subprogram implementation helloworld.i
properties
  Source_Language => C;
  Source_Text => "helloworld.c";
  Source_Name => "user_helloworld";
end helloworld.i
```

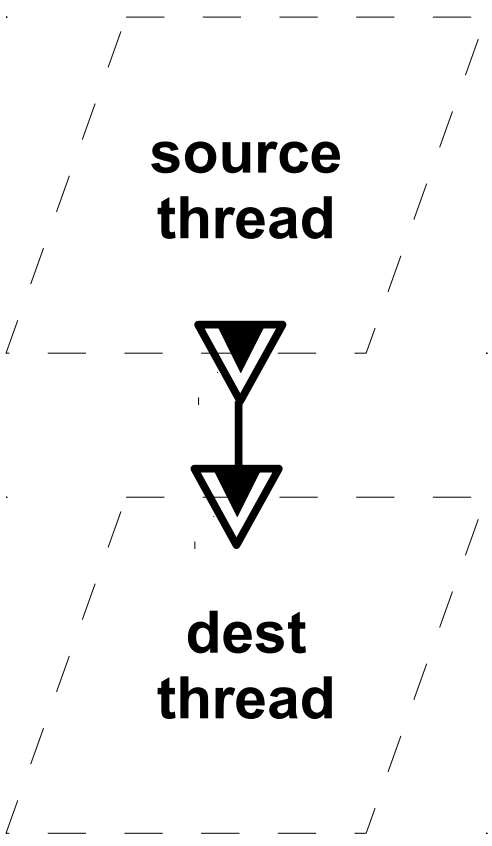
subprogram




```
thread implementation arincprocess.i
calls
  Mycall : {pspg:subprogram helloworld.i;};
properties
  Period => 10ms;
  Compute_Execution_Time => 0ms .. 10ms;
  Deadline => 10ms;
end arincprocess.i;
```

Intra-partition - buffer

- **Use AADL event data ports**
  - **Data queuing**
  - **Connection across AADL threads (ARINC653 processes)**



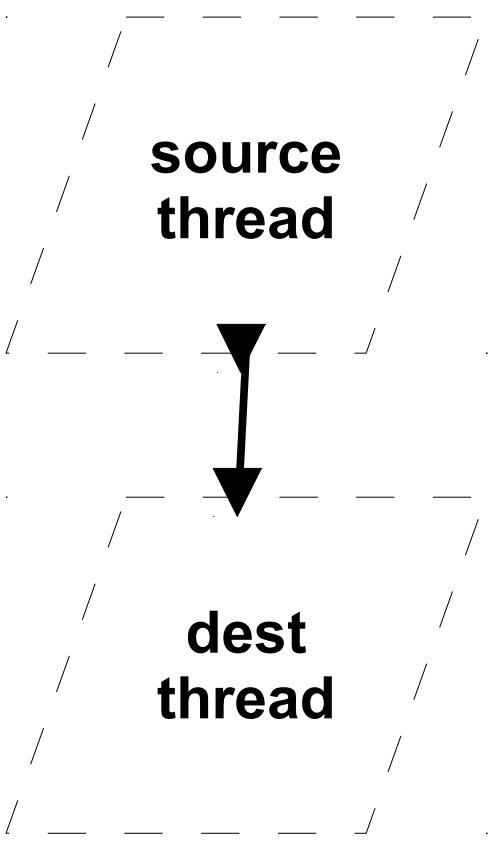

```
thread src
features
  data_source : out event data port mytype.i;
end src;

thread dst
features
  data_sink : in event data port mytype.i;
end dst;

process implementation partitioncontent.i
subcomponents
  thr_src : thread implementation src.i;
  thr_dst : thread implementation dst.i;
connections
  port thr_src.data_source → thr_dst.data_sink;
end partitioncontent.i;
```

Intra-partition - blackboard

- **Use AADL event data ports**
  - **No queuing, new data instances replace old ones**
  - **Connection across AADL threads (ARINC653 processes)**



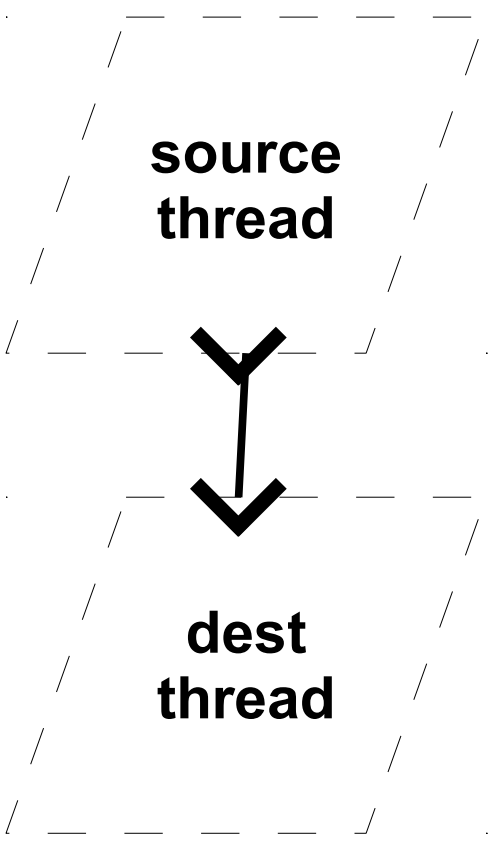

```
thread src
features
  data_source : out data port mytype.i;
end src;

thread dst
features
  data_sink : in data port mytype.i;
end dst;

process implementation partitioncontent.i
subcomponents
  thr_src : thread implementation src.i;
  thr_dst : thread implementation dst.i;
connections
  port thr_src.data_source → thr_dst.data_sink;
end partitioncontent.i;
```

Intra-partition - events

- **Use AADL event ports**
  - **Events processing, no queuing**
  - **Connection across AADL threads (ARINC653 processes)**



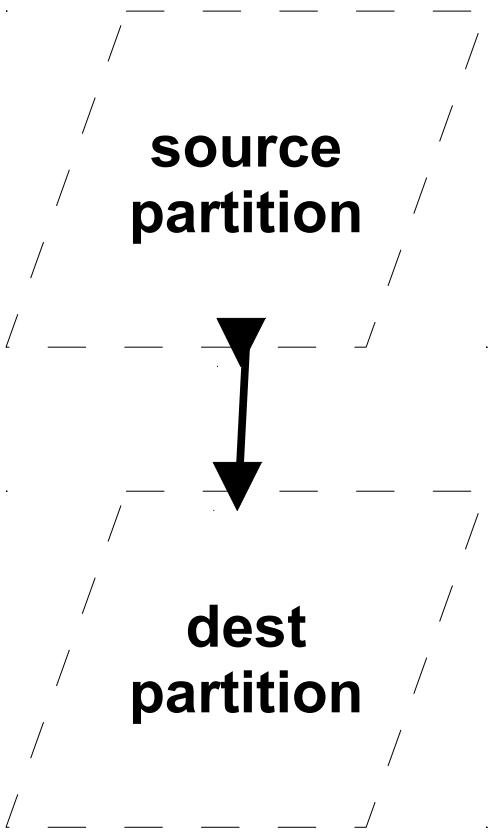

```
thread src
features
  data_source : out event port mytype.i;
end src;

thread dst
features
  data_sink : in event port mytype.i;
end dst;

process implementation partitioncontent.i
subcomponents
  thr_src : thread implementation src.i;
  thr_dst : thread implementation dst.i;
connections
  port thr_src.data_source → thr_dst.data_sink;
end partitioncontent.i;
```

Inter-partition - sampling

- **Use AADL data ports**
  - **NO queuing, new data instances replace old ones**
  - **Connection across AADL process (ARINC653 partitions)**



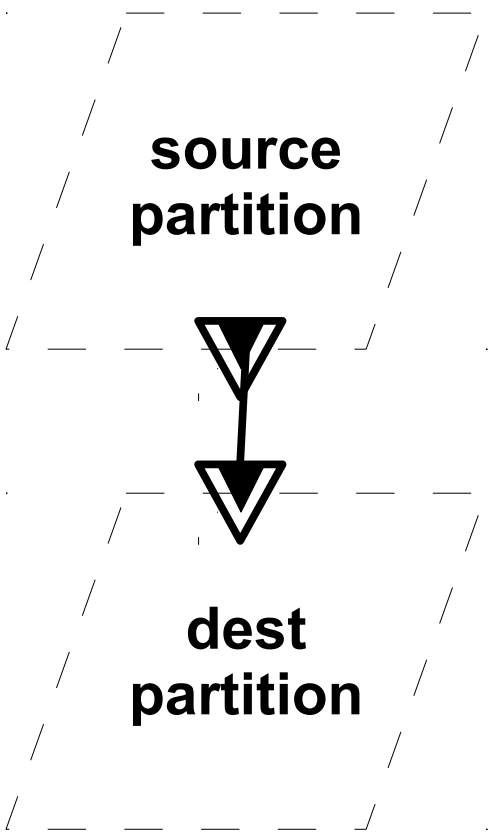

```
process src
features
  data_source : out data port mytype.i;
end src;

process dst
features
  data_sink : in data port mytype.i;
end dst;

system implementation arincsystem.i
subcomponents
  part_src : process implementation src.i;
  part_dst : process implementation dst.i;
connections
  port part_src.data_source → part_dst.data_sink;
end arincsystem.i;
```

Inter-partition - queuing

- **Use AADL event data ports**
  - **Data queuing**
  - **Connection across AADL process (ARINC653 partitions)**




```
process src
features
  data_source : out event data port mytype.i;
end src;

process dst
features
  data_sink : in event data port mytype.i;
end dst;

system implementation arincsystem.i
subcomponents
  part_src : process implementation src.i;
  part_dst : process implementation dst.i;
connections
  port part_src.data_source → part_dst.data_sink;
end arincsystem.i;
```

Communication specificities

a partitioned pok operating system

- Queuing policy
  - ARINC653::Queuing_Policy (ARINC653-specific)
  - FIFO or By_Priority values
- Queuing pool size
  - Queue_Size (standard property)
- Timeout
  - ARINC653::Timeout property (ARINC653-specific)

Health Monitoring

a partitioned pok operating system

- **ARINC653::HM_Errors**
  - **Associated with AADL processor, virtual processor or thread**
  - **Describe errors that may be raised in each level**
- **ARINC653::HM_Actions**
  - **Associated with AADL processor, virtual processor or thread**
  - **Associate a recovery action for each potential error**

ARINC653 system validation

- **System consistency**
  - **Verify architecture correctness**
- **Health Monitoring policy impacts**
  - **Can a LEVEL_B (low-critical) partition impacts a LEVEL_A partition (higher criticality ?)**
  - **Analysis using partitions AND process informations**

System validation – cont'd

a partitioned pok operating system

- **Scheduling analysis**
  - **Space isolation**
  - **Partitions scheduling**
  - **Use module AND partitions scheduling properties**
  - **See Cheddar**
    http://beru.univ-brest.fr/~singhoff/cheddar/

AADL modeling, summary

a partitioned pok operating system

- **Describe the whole architecture**
  - **Everything, except application concerns**
  - **Specify architecture requirements & properties**
- **Suitable to drive the development process**
  - **Validation & analysis**
  - **Configuration & code generation**
  - **Automatic certification**

Outline

a partitioned pok operating system

- Reminder about partitioned architectures
- Introduction to XML and AADL formalisms
- AADL specification and validation
- **XML specification and validation**

XML - Module mapping

```
<ARINC_653_Module
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=""
  ModuleName="MyARINCModule"
  ModuleVersion="22-March-2010">
<!-- Content of the module (XML subitems) →
</ARINC_653_Module>
```

- **One XML item that contains everything**
  - **Partitions, connections, … are described in subitems**

XML - Partition mapping

```
<Partition PartitionIdentifier="1"
 PartitionName="myfirstpartition"
 Criticality ="LEVEL_B"
 SystemPartition="false" EntryPoint="main">
 <!-- subitems describe partitions inter-partitions ports -->
</Partition>
```

Name of the component

ARINC653::Criticality property

- One dedicated node
  - Contain inter-partitions ports description
- Information disseminated in module subitems
  - Time isolation
  - Space isolation
- Missing informations
  - Process management
  - Intra-partition ports

XML – Specification of Inter-partitions ports

a partitioned pok operating system

Size of the data associated with the port

Name of the AADL interface (port)

```
<Partition PartitionIdentifier="1"
 PartitionName="myfirstpartition" Criticality ="LEVEL_B"
 SystemPartition="false" EntryPoint="main">

  <Sampling_Port PortName="Act_1Ds" MaxMessageSize="20"
   Direction="DESTINATION" RefreshRateSeconds="0.100"/>

<Queuing_Port PortName="Stat_5Sq" MaxMessageSize="30"
   Direction="SOURCE" MaxNbMessages="30"/>
</Partition>
```

AADL out port

AADL Queue_Size property

XML - Inter-partitions ports connection

a partitioned pok operating system

```
<Connection_Table>
  <Channel ChannelIdentifier="1" ChannelName="achannel">
    <Source>
      <Standard_Partition PartitionIdentifier=”1”
       PartitionName="myfirstpartition" PortName="data_source"/>
    </Source>
    <Destination>
      <Standard_Partition PartitionIdentifier="2"
       PartitionName="myotherpartition" PortName="data_sink"/>
    </Destination>
  </Channel>
  <!-- description of other channels →
</Connection_Table>
```

XML specification of partitions scheduling policy

a partitioned pok operating system

Partitions scheduling policy repeated every 0.2s

```
<Module_Schedule MajorFrameSeconds="0.200">
  <Partition_Schedule PartitionIdentifier="1"
   PartitionName="myfirstpartition"
   PeriodSeconds="0.100" PeriodDurationSeconds="0.020">
     <Window_Schedule WindowIdentifier="101" WindowStartSeconds="0.0"
      WindowDurationSeconds="0.020" PartitionPeriodStart="true"/>
     <Window_Schedule WindowIdentifier="102" WindowStartSeconds="0.1"
      WindowDurationSeconds="0.020" PartitionPeriodStart="true"/>
  </Partition_Schedule>
  <!-- other partition_schedule items -->
</Module_Schedule>
```

Scheduling of partition 1 changes every 0.1s … and uses 20ms

XML – Space isolation (memory allocation)

a partitioned pok operating system

```
<Partition_Memory PartitionIdentifier="1" PartitionName="firstpart">
  <Memory_Requirements Type="CODE" SizeBytes="20000"
Access="READ_ONLY"/>
  <Memory_Requirements Type="DATA" SizeBytes="20000"
Access="READ_WRITE"/>
  <Memory_Requirements Type="INPUT_OUTPUT" SizeBytes="128000"
   PhysicalAddress="FF000000" Access="READ_WRITE"/>
</Partition_Memory>
<!-- description of other memory allocation -->
```

- **Memory requirements for each partition**
  - **Specify memory location and type**
  - **Can be deduced from AADL models from memory bindings**

XML Health Monitoring specification – system table

a partitioned pok operating system

- Information about the whole HM policy
  - Errors raised
  - Recovery location
- Do not specify the recovery policy
  - Module recovery policy in Module_HM_Table
  - Partition recovery policy in Partition_HM_Table
  - Process recovery policy not specified

XML Health Monitoring system table example

a partitioned pok operating system

```
<System_HM_Table>
  <System_State_Entry SystemState="1" Description="ModuleInit">
     <Error_ID_Level ErrorIdentifier="1"
      Description="Configuration Error" ErrorLevel="MODULE"/>
     <!-- description of other errors -->
  </System_State_Entry>
  <System_State_Entry SystemState="4" Description="PartitionInit">
     <Error_ID_Level ErrorIdentifier="3" Description="partition
config error" ErrorLevel="PARTITION"/>
    <!-- description of other errors -->
  </System_State_Entry>
  <System_State_Entry SystemState="7" Description="ProcessExecution">
    <Error_ID_ Level ErrorIdentifier="5"
     Description="segmentation error" ErrorL evel="PROCESS"
     ErrorCode="MEMORY_VIOLATION"/>
    <!-- description of other errors -->
  </System_State_Entry>
<!-- describe Health Monitoring policy for other states →
</System_HM_Table>
```

XML Health Monitoring module table

a partitioned pok operating system

```
<Module_HM_Table ModuleCallback="module_HM_callback">
  <System_State_Entry SystemState="1" Description="module init">
    <Error_ID_Action ErrorIdentifier="1" A ction="SHUTDOWN"/>
    <Error_ID_Action ErrorIdentifier="2" Action="SHUTDOWN"/>
    <Error_ID_Action ErrorIdentifier="5" Action="SHUTDOWN"/>
    <Error_ID_Action ErrorIdentifier="6" Action="IGNORE"/>
    <Error_ID_Action ErrorIdentifier="7" Action="IGNORE"/>
  </System_State_Entry>
  <!-- describe HM recovery policy for other systems →
</Module_HM_Table>
```

- **Specify recovery strategy for each state**
  - **Associate actions for each error**
  - **Handle only module-level errors !**
  - **Partition level errors described in the Partition_HM_Table**
  - **Can be deduced from AADL properties in processor components**

XML Health Monitoring partition table

a partitioned pok operating system

```
<Partition_HM_Table PartitionIdentifier="1" PartitionName="firstpart"
 PartitionCallback="partition_HM_callback">
  <System_State_Entry SystemState="7"
   Description="process execution">
    <Error_ID_Action ErrorIdentifier="5" Description="segmentation
error" Action="IDLE"/>
    <Error_ID_Action ErrorIdentifier="6" Description="time duration
exceeded" Action="WARM_START"/>
    <!-- description of other fault and actions -->
  </System_State_Entry>
  <!-- description of the HM policy for the other states -->
</Partition_HM_Table>
```

- **Specify recovery strategy for each state**
  - **Associate actions for each error**
  - **Handle only partition-level errors !**
  - **Can be deduced from AADL models using virtual processors**

XML validation

a partitioned pok operating system

- **System consistency**
  - **Check memory isolation**
  - **Verify channels consistency**
  - **Limited validation, need more informations**
- **Scheduling**
  - **Partitions scheduling**
  - **Cannot validation scheduling for each partition**
- **Limited analysis**
  - **Lack of specification**

Outline

a partitioned pok operating system

- Reminder about partitioned architectures
- Introduction to XML and AADL formalisms
- AADL specification and validation
- XML specification and validation

XML file generation

a partitioned pok operating system

- **AADL for architecture modeling**
  - **Better analysis & validation**
  - **Suitable for code generation**
- **XML file generation**
  - **Available in Ocarina AADL toolsuite**
  - **Deployment and interoperability purposes**

Conclusion

a partitioned pok operating system

- **AADL to address the whole development process**
  - **Complete system description**
  - **System validation, code generation, …**
  - **User & tool-friendly !**
  - **Still a newcomer**
- **XML**
  - **Very specific to each O/S implementation**
  - **Redundant information, not user-friendly**
  - **Limited in analysis, validation & code generation**

Questions ?