Understanding the Relationship between AADL and Real-Time Embedded Systems Operating Systems

Joyce L Tokar, PhD
AS-2C Committee Member, Co-editor, Programming Language Annex author
Pyrrhus Software
480-951-1019
tokar@pyrrhusoft.com
Overview

• Representing Operating Systems in an AADL Specification.
• The AADL Execution Environment.
• Future Directions with AADL.
RTOS Representations

- OS is part of a processor specification.
- OS can be modeled using software and execution platform components.
  - OS memory may be modeled as a process with access to the actual memory.
  - OS may be represented as a thread which is a subcomponent of the OS process.
AADL Processors

• An abstraction of hardware and software that is responsible for scheduling and executing threads.
• Execute threads declared in application software systems.
• Execute threads declared in devices that can be accessed from the processor.
• May contain memories and may access memories and devices via buses.

<table>
<thead>
<tr>
<th>Features:</th>
<th>Subcomponents:</th>
</tr>
</thead>
<tbody>
<tr>
<td>server subprogram</td>
<td>memory</td>
</tr>
<tr>
<td>port</td>
<td></td>
</tr>
<tr>
<td>Requires bus access</td>
<td>Connections: no</td>
</tr>
<tr>
<td></td>
<td>Modes: yes</td>
</tr>
</tbody>
</table>
AADL Processes

- Represents a virtual address space.
- A complete implementation of a process must contain at least one thread or thread group subcomponent.

<table>
<thead>
<tr>
<th>Features:</th>
<th>Subcomponents:</th>
</tr>
</thead>
<tbody>
<tr>
<td>server subprogram</td>
<td>data</td>
</tr>
<tr>
<td>port</td>
<td>thread</td>
</tr>
<tr>
<td>port groups</td>
<td>thread group</td>
</tr>
<tr>
<td>Provides data access</td>
<td>Connections: yes</td>
</tr>
<tr>
<td>Requires data access</td>
<td>Modes: yes</td>
</tr>
</tbody>
</table>
AADL Threads

- Represents a sequential flow of control.
- Models a schedulable unit that transitions between various scheduling states.
- Always executes within the virtual address space of a process.

<table>
<thead>
<tr>
<th>Features:</th>
</tr>
</thead>
<tbody>
<tr>
<td>server subprogram</td>
</tr>
<tr>
<td>port</td>
</tr>
<tr>
<td>port groups</td>
</tr>
<tr>
<td>Provides data access</td>
</tr>
<tr>
<td>Requires data access</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subcomponents:</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connections:</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
</tr>
</tbody>
</table>
AADL Scheduling Properties

Activate_Deadline: Time
Activate_Execution_Time: Time_Range
Activate_Entrypoint: aadlstring
Active_Thread_Queue_Handling_Protocol: inherit enumeration (flush, hold)
AllowedDispatch Protocol: list of SupportedDispatch Protocols
Allowed_Period: list of Time_Range
Compute_Deadline: Time
Compute_Entrypoint: aadlstring
Compute_Execution_Time: Time_Range
Concurrency_Control_Protocol: SupportedConcurrency_Control_Protocols
Deactivate_Deadline: Time
Deactivate_Execution_Time: Time_Range
Deactivate_Entrypoint: aadlstring
Deadline: Time
Device_Dispatch_Protocol: Supported_Dispatch_Protocols
Dispatch Protocol: Supported_Dispatch_Protocols
....
Processor Provides OS

System Demo

Process P2
Thread T5
Thread T6

Process P1
Thread T1
Thread T2

Mission Processor
AADL Models OS

System DRTOS

OS Process

Scheduler

data
Overview

• Representing Operating Systems in an AADL Specification.
• The AADL Execution Environment.
  ▪ Tools
  ▪ Application Software Guidelines
• Future Directions with AADL.
AADL Development Environment

AADL-Based Software & Systems Integration Toolset

- software tools
- target hardware specifications
- re-engineering of legacy software
- application development tools
- traditional development
- other specialized tools

Complete, Validated Executable System
AADL Execution Environment

**AADL Executive/Kernel**
Execution control, timing control, data synchronization, interprocess communication, mode change, fault recovery

**Operating Environment**
Operating system, run-time system, customer kernel, etc.

**Embedded Hardware Target**
PPC, x86, custom platform
Programming Language Guidelines

### AADL Software Components

<table>
<thead>
<tr>
<th>AADL</th>
<th>Ada 95</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>Subprogram</td>
<td>Procedure, Function</td>
<td>Procedure, Function</td>
</tr>
<tr>
<td>Thread</td>
<td>Subprogram</td>
<td>Subprogram</td>
</tr>
<tr>
<td>Thread Group</td>
<td>Subprograms &amp; data in a package</td>
<td>Subprogram &amp; data in an include file</td>
</tr>
<tr>
<td>Process</td>
<td>Application</td>
<td>Application</td>
</tr>
</tbody>
</table>
**Programming Language Guidelines**

**Communication and Control**

<table>
<thead>
<tr>
<th>AADL</th>
<th>Ada 95</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Port</td>
<td>Static Variables in Packages</td>
<td>Static variables in include files</td>
</tr>
<tr>
<td>Event Port</td>
<td>Parameter to Raise_Event</td>
<td>Parameter to Raise_Event</td>
</tr>
<tr>
<td>Event Data Port</td>
<td>Record with data pointer; Raise_Event</td>
<td>Struct with data pointer; Raise_Event</td>
</tr>
<tr>
<td>Subprogram Parameter</td>
<td>Formal Parameters</td>
<td>Formal Parameters</td>
</tr>
<tr>
<td>Requires Provides</td>
<td>Shared data / protected object</td>
<td>Shared data</td>
</tr>
</tbody>
</table>
# Programming Language Guidelines

## Packages and Libraries

<table>
<thead>
<tr>
<th>Package</th>
<th>Package</th>
<th>Include file</th>
<th>Private Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>AADL</td>
<td>Ada 95</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>Package</td>
<td>Package</td>
<td>Include file</td>
<td>Private Macro</td>
</tr>
<tr>
<td>Private Part</td>
<td>Private Part</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
package Sampling
public
data Sample
  properties Source_Data_Size => 16 B;
end Sample;

data Sample_Set
  properties Source_Data_Size => 1 MB;
end Sample_Set;

data Dynamic_Sample_Set extends Sample_Set
end Dynamic_Sample_Set;
end Sampling;
package Sampling is
  type Sample is record
    Byte0 : character;  Byte1 : character;  Byte2 : character;
    Byte3 : character;  Byte4 : character;  Byte5 : character;
    Byte6 : character;  Byte7 : character;  Byte8 : character;
    Byte9 : character;  ByteA : character;  ByteB : character;
    ByteC : character;  ByteD : character;  ByteE : character;
    ByteF : character;
  end record;
  for Sample'Size use 16 * 8;
-- Note that Ada 95 Size is represented in bits. The AADL property was for
-- a size of 16 bytes, hence the * 8.

  type Sample_Set is array ( 1..65536 ) of Sample;
  for Sample_Set'Size use 1_048_576 * 8; -- 1MB == 1048576 bytes

  subtype Dynamic_Sample_Set is Sample_Set;
  type Access_Sample_Set is access all Sample_Set;
end Sampling;
Package - C

/* sampling.h */

typedef struct {
    char Byte0, Byte1, Byte2, Byte3, Byte4, Byte5, Byte6, Byte7,
    Byte8, Byte9, ByteA, ByteB, ByteC, ByteD, ByteE, ByteF;
} Sample;

typedef Sample Sample_Set[65535];
typedef Sample_Set Dynamic_Sample_Set;
typedef Sample_Set *Access_Sample_Set;
Programming Language Guidelines

AADL Specification

process Sample_Manager
  features
    Input_Sample: in event data port Sampling::Sample;
    External_Samples: requires data access Sampling::Sample_Set;
    Result_Sample: out event data port Sampling::Sample;
end Sample_Manager
Programming Language Guidelines

**AADL Specification**

```aadl
process implementation Sample_Manager.Slow_Update
subcomponents
  Samples: data Sampling::Sample_Set;
  Init_Samples : thread Init_Samples;
  Collect_Samples: thread Collect_Samples.Batch_Update;
  Distribute: thread Distribute_Samples ;
connections
  data access Samples -> Init_Samples.SampleSet;
  data access External_Samples -> Init_Samples.OrigSet;
  data access Samples -> Collect_Samples.SampleSet;
  event data port Input_Sample -> Collect_Samples.Input_Sample;
  data access Samples -> Distribute.SampleSet;
  event data port Distribute.UpdatedSamples -> Result_Sample;
end Sample_Manager.Slow_Update ;
```
package Sample_Manager_Slow_Update_Ports is

  type Input_Sample_Event_Data_Port is
    record
      Data : Sampling.Access_Sample_Set;
      Event : AADL.Event := AADL."+"( "Input Samples" );
    end record;

  type Result_Sample_Event_Data_Port is
    record
      Data : Sampling.Access_Sample_Set;
      Event : AADL.Event := AADL."+"( "Updated Samples" );
    end record;

  Input_Sample : Input_Sample_Event_Data_Port;
  External_Samples : Sampling.Access_Sample_Set;
  Result_Sample : Result_Sample_Event_Data_Port;
  Samples : aliased Sampling.Sample_Set;

  procedure Initialize;

end Sample_Manager_Slow_Update_Ports;
Ports - C

/* slow_update_ports.h */

#include "../aadl/AADL.h"
#include "sampling.h"

typedef struct {
    Access_Sample_Set EDP_Data;
    Event EDP_Event;
} Input_Sample_Event_Data_Port;

typedef struct {
    Access_Sample_Set EDP_Data;
    Event EDP_Event;
} Result_Sample_Event_Data_Port;
with AADL;
with Sample_Manager_Slow_Update_Ports;

procedure Distribute_Samples(
  SampleSet : Sampling.Access_Sample_Set;
  UpdatedSamples : out
Sample_Manager_Slow_Update_Ports.Result_Sample_Event_Data_Port )

is

type Distribute_Samples_Parameter_Block is record
  SampleSet : Sampling.Access_Sample_Set;
  UpdatedSamples : Sample_Manager_Slow_Update_Ports.Result_Sample_Event_Data_Port;
end record;

Distribute_Samples_Parameters : Distribute_Samples_Parameter_Block
  := ( SampleSet, UpdatedSamples );
procedure Distribute_Samples_Thread is
begin
  loop
    AADL.Await_Dispatch;
    -- do work
    Distribute_Samples_Parameters.UpdatedSamples.Data :=
      Distribute_Samples_Parameters.SampleSet;
    AADL.Raise_Event(
      Distribute_Samples_Parameters.UpdatedSamples.Event );
  end loop;
end Distribute_Samples_Thread;

begin -- Distribute_Samples
  AADL.Create_Thread( Distribute_Samples_Thread'Address,
                      Distribute_Samples_Parameters'Address );
end Distribute_Samples;
/* threads.c */
#include "slow_update_ports.h"

void Init_Samples(void *arg) {
    Parameters *Parms = (Parameters *)arg;
    Access_Sample_Set OrigSet = (Access_Sample_Set)Parms->P1;
    Access_Sample_Set Sample_Set = (Access_Sample_Set)Parms->P2;

    while (1) {
        Await_Dispatch();
        /* do work */
    }
}

void Collect_Samples(void *arg) {
    Parameters *Parms = (Parameters *)arg;
    Input_Sample_Event_Data_Port *InSample =
        (Input_Sample_Event_Data_Port *)Parms->P1;
    Access_Sample_Set Sample_Set = (Access_Sample_Set)Parms->P2;

    while (1) {
        Await_Dispatch();
        /* do work */
    }
}
void Distribute_Samples(void *arg) {
    Parameters *Parms = (Parameters *)arg;
    Access_Sample_Set SampleSet = (Access_Sample_Set)Parms->P1;
    Result.Sample.Event>Data.Port *UpdatedSamples = 
    (Result.Sample.Event>Data.Port *)Parms->P2;

    while (1) {
        Await_Dispatch();
        /* do work */
        UpdatedSamples->EDP_Data = SampleSet;
        Raise_Event(UpdatedSamples->EDP_Event);
    }
}
with AADL; with Sampling;
with Sample_Manager_Slow_Update_Ports; with Init_Samples;
with Collect_Samples; with Distribute_Samples;

procedure Slow_Update_Process is
  procedure Init_Samples is new Init_Samples;
  procedure Batch_Update is new Collect_Samples;
  procedure Distribute is new Distribute_Samples;

begin
  -- Initialize ports and threads
  Sample_Manager_Slow_Update_Ports.Initialize;
  Init_Samples( Sample_Manager_Slow_Update_Ports.External_Samples,
                Sample_Manager_Slow_Update_Ports.Samples'access );
  Batch_Update( Sample_Manager_Slow_Update_Ports.Input_Sample,
                Sample_Manager_Slow_Update_Ports.Samples'access );
  Distribute( Sample_Manager_Slow_Update_Ports.Samples'access,
              Sample_Manager_Slow_Update_Ports.Result_Sample );

  loop
    AADL.Await_Dispatch;
    -- do work
  end loop;
end Slow_Update_Process;
void Init_Samples(void *);
void Collect_Samples(void *);
void Distribute_Samples(void *);
void Initialize(void);

void Slow_Update_Process(void) {
    Input_Sample_Event_Data_Port Input_Sample = { 0, "Input Samples" };
    Access_Sample_Set External_Samples;
    Result_Sample_Event_Data_Port Result_Sample = { 0, "Updated Samples" };
    Sample_Set Samples;

    Parameters Init_Parms = { External_Samples, Samples };
    Parameters Collect_Parms = { &Input_Sample, Samples };
    Parameters Distribute_Parms = { Samples, &Result_Sample };

    Initialize();

    Create_Thread( Init_Samples, &Init_Parms );
    Create_Thread( Collect_Samples, &Collect_Parms );
    Create_Thread( Distribute_Samples, &Distribute_Parms );

    while (1) {
        Await_Dispatch();
        /* do work */
    }
}
procedure Date_Book is
  type Date is
      record
        Day : Integer range 1 .. 31;
        Mnth : Month_Name;
        Year : Integer range 0 .. 4000;
      end record;
  ...
  Tomorrow, Yesterday : Date;
  ...
end Date_Book;
AADL

data Date

properties
    Type_Source_Name => "Date_Book.Date";
end Date;

data implementation Date.others
subcomponents
    Day : data basic::integer;
    Mnth : data basic::integer;
    Year : data basic::integer;
end Date.others;
AADL

```plaintext
system Date_Book
end Date_Book;

system implementation Date_Book.imp1
subcomponents
  Tomorrow : data Date.others;
  Today : data Date.others;
end Date_Book.imp1;
```
package AADL is
  type Event is access String;
  function "+" (S : String) return Event;

  type Error is access String;
  function "+" (S : String) return Error;

  procedure Create_Thread( Thread_Name : System.Address;
                           Thread_Parameters : System.Address);
  procedure Await_Dispatch;

  procedure Get_Resource;
  procedure Release_Resource;

  procedure Raise_Event( EVID : in Event );
  procedure Raise_Error( ERID : in Error );

  procedure Dispatch_Status;
  procedure Connection_Status;

  procedure Stop_Process;
  procedure Abort_Process;

  procedure Stop_Processor;
  procedure Abort_Processor;
  procedure Stop_System;
  procedure Abort_System;
end AADL;
/* AADL.h */

typedef char *Event;
typedef struct { void *P1; void *P2; } Parameters;

void Create_Thread( void Thread(void *Arg), void *Parms );
void Await_Dispatch( void );

void Get_Resource( void );
void Release_Resource( void );

void Raise_Event( Event EVID );
void Raise_Error( Event IRIS );

void Dispatch_Status( void );
void Connection_Status( void );

void Stop_Processor( void );
void Abort_Processor( void );

void Stop_System( void );
void Abort_System( void );
Overview

- The AADL Execution Environment.
- Future Directions with AADL.
Partitioned RTOSes
ARINC 653

• Represented by software and execution platform components.
• Investigating the need for the introduction of new components to represent partitioned operating systems.
• Investigating the definition of new properties targeted toward partitioned operating systems.
AADL Runtime Environment

- AADL Runtime Executive
- Application/AADL API layer
- AADL generator
Thank You

aadl.info