Architecture Analysis and Design Language
(AADL)

Part 3
**Component Type and Implementation: Flows**

**Flow** is a path through an architecture (components and connections).

**Flow specification** specifies how data and/or control flows from input to output ports through a sequence of components and connections.

Flow specification defines different paths taken for transporting data from its creation to its consumption. It is the chain of elements that create, transport, process, and eventually consume the data.

Flow specifications are specified in the component type by showing how data flows in the external interfaces (features).

**Flow implementation** explains how data flows through the different subcomponents.

Flow specifications are later refined in the component implementations (flow implementations).

**End-to-end flow** supports analysis, e.g. end-to-end timing and latency.

Flow implementation is the outside view of a flow through a component. It is a realization of a flow specification.

End-to-end flow is a logical flow from a source through other components and connections to a destination.
Flow specification specifies how data and/or control flows from input to output ports through a sequence of components and connections. Flow implementation explains how data flows through the different subcomponents.

Define flow(s) for each of the components above.
Threads
Threads

```
thread ComputeDesiredSpeed
  features
    actual_velocity: in data port;
    output_velocity: out data port;
  flows
    flow_M_T: flow path actual_velocity -> output_velocity;
  properties
    Dispatch_Protocol => Periodic;
    Period => 10Ms;
    Compute_Execution_Time => 2Ms .. 2Ms;
end ComputeDesiredSpeed;
```
Define flow(s) for ScanInputPorts and Compute Throttle Setting threads.
Threads

thread ComputeDesiredSpeed
features
  actual_velocity: in data port;
  output_velocity: out data port;
flows
flow_W_T: flow path actual_velocity -> output_velocity;
properties
  Dispatch_Protocol => Periodic;
  Period => 10 ms;
  Compute_Execution_Time => 2 ms .. 2 ms;
end ComputeDesiredSpeed;

thread ComputeThrottleSetting
features
  output_velocity: in data port;
  brake_pedal: in data port;
  throttle_setting: out data port;
flows
flow_W_T: flow path output_velocity -> throttle_setting;
flow_B_T: flow path brake_pedal -> throttle_setting;
properties
  Dispatch_Protocol => Periodic;
  Period => 5 ms;
  Compute_Execution_Time => 3 ms .. 3 ms;
end ComputeThrottleSetting;

process CruiseControlProcess
features
  wheel_pulses: in data port;
  brake_pedal: in data port;
  throttle_setting: out data port;
flows
flow_W_T: flow path wheel_pulses -> throttle_setting;
flow_B_T: flow path brake_pedal -> throttle_setting;
end CruiseControlProcess;
Flow specification specifies how data and/or control flows from input to output ports through a sequence of components and connections. Flow implementation explains how data flows through the different subcomponents.

Define flow(s) for the component above.
Define flow(s) for CruiseControlProcess.
Process

```
process CruiseControlProcess
  features
    wheel_pulses: in data port;
    brake_pedal: in data port;
    throttle_setting: out data port;
  flows
    Flow_W_T: flow path wheel_pulses -> throttle_setting;
    Flow_B_T: flow path brake_pedal -> throttle_setting;
  end CruiseControlProcess;
```
Flow specification specifies how data and/or control flows from input to output ports through a sequence of components and connections.

Flow implementation explains how data flows through the different subcomponents.

Define flow(s) for the component above.
Flow specification specifies how data and/or control flows from input to output ports through a sequence of components and connections.

Flow implementation explains how data flows through the different subcomponents.

Define flow(s) for the component above.
Process and Threads
Process and Threads

Implementation

brake_pedal  d5  scan_input_ports.flow_B_T  d3  compute_throttle_setting.flow_B_T  d6  throttle_setting

Flow_B_T: flow path brake_pedal -> throttle_setting;

Flow_B_T: flow path brake_pedal -> d5 -> scan_input_ports.flow_B_T -> d3 -> compute_throttle_setting.flow_B_T -> d6 -> throttle_setting;
Process and Threads

Define flow(s) for the component above.
Process and Threads

flows
Flow_B: Flow path brake_pedal -> d5 -> scan_input_ports.B.T -> d3 -> compute_throttle_setting.B.T -> d6 -> throttle_setting;
**Flow specification** specifies how data and/or control flows from input to output ports through a sequence of components and connections.

**Flow implementation** explains how data flows through the different subcomponents.

Define flow(s) for the component above.
Define flow(s) for the components above.
System

```plaintext
system CruiseControl
  features
    cc_ui: in data port;
    cc_ud: out data port;
    cc_wrs: in data port;
    cc_bp: in data port;
    cc_ta: out data port;
  flows
    Flow_W_T: flow path cc_wrs -> cc_ta {
      Latency => 20ms .. 200ms;
    };
    Flow_B_T: flow path cc_bp -> cc_ta {
      Latency => 15ms .. 150ms;
    };
end CruiseControl;
```
Flow specification specifies how data and/or control flows from input to output ports through a sequence of components and connections.

Flow implementation explains how data flows through the different subcomponents.

Define flow(s) for the component above.
System

Implementation

system implementation CruiseControl.impl
subcomponents
    cc_process: process CThreads::CruiseControlProcess.impl;
connections
d2: port cc_bp -> cc_process.brake_pedal;
d1: port cc_wrs -> cc_process.wheel_pulses;
d3: port cc_process.throttle_setting -> cc_ta;
flows
    Flow_W_T: flow path cc_wrs -> d1 -> cc_process.Flow_W_T -> d3 -> cc_ta;
    Flow_B_T: flow path cc_bp -> d2 -> cc_process.Flow_B_T -> d3 -> cc_ta;
end CruiseControl.impl;
Flow source specifies the origination of the flow.
Flow sink specifies the termination of the flow.

Define flow(s) for the components above.
Top-Level System

device ThrottleActuator
  features
    ta_cc: in data port;
    flows
ta_sink: flow sink ta_cc {Latency => 15Ms .. 15Ms;}
  properties
    SEI::NetWeight => 2.0kg;
end ThrottleActuator;

device WheelRotationSensor
  features
    wrs_cc: out data port;
    flows
    wrs_source: flow source wrs_cc {
      Latency => 5Ms .. 5Ms;
    }
  properties
    SEI::NetWeight => 0.25kg;
end WheelRotationSensor;

device BrakePedal
  features
    bp_cc: out data port;
    flows
    bp_source: flow source bp_cc {
      Latency => 7Ms .. 7Ms;
    }
  properties
    SEI::NetWeight => 0.25kg;
end BrakePedal;
End-to-end flow is a logical flow from a source through other components and connections to a destination.

Define end-to-end flow(s) for the components above.
Top-Level System
Flow Latency

- Flow latency is the amount of time it takes for information to flow from the starting point of an end-to-end flow to its destination via connections and possibly intermediate components.

- The starting point, intermediate components, and destination can be threads and devices.

- Flow latency is affected by several factors:
  - Processing time by tasks in the end-to-end flow. Tasks are AADL threads and AADL devices.
  - Processing delay due to queuing or sampling.
  - Transfer time of information between tasks along connections.
  - Transfer delay due to queuing or waiting for time slots in the transfer protocol.