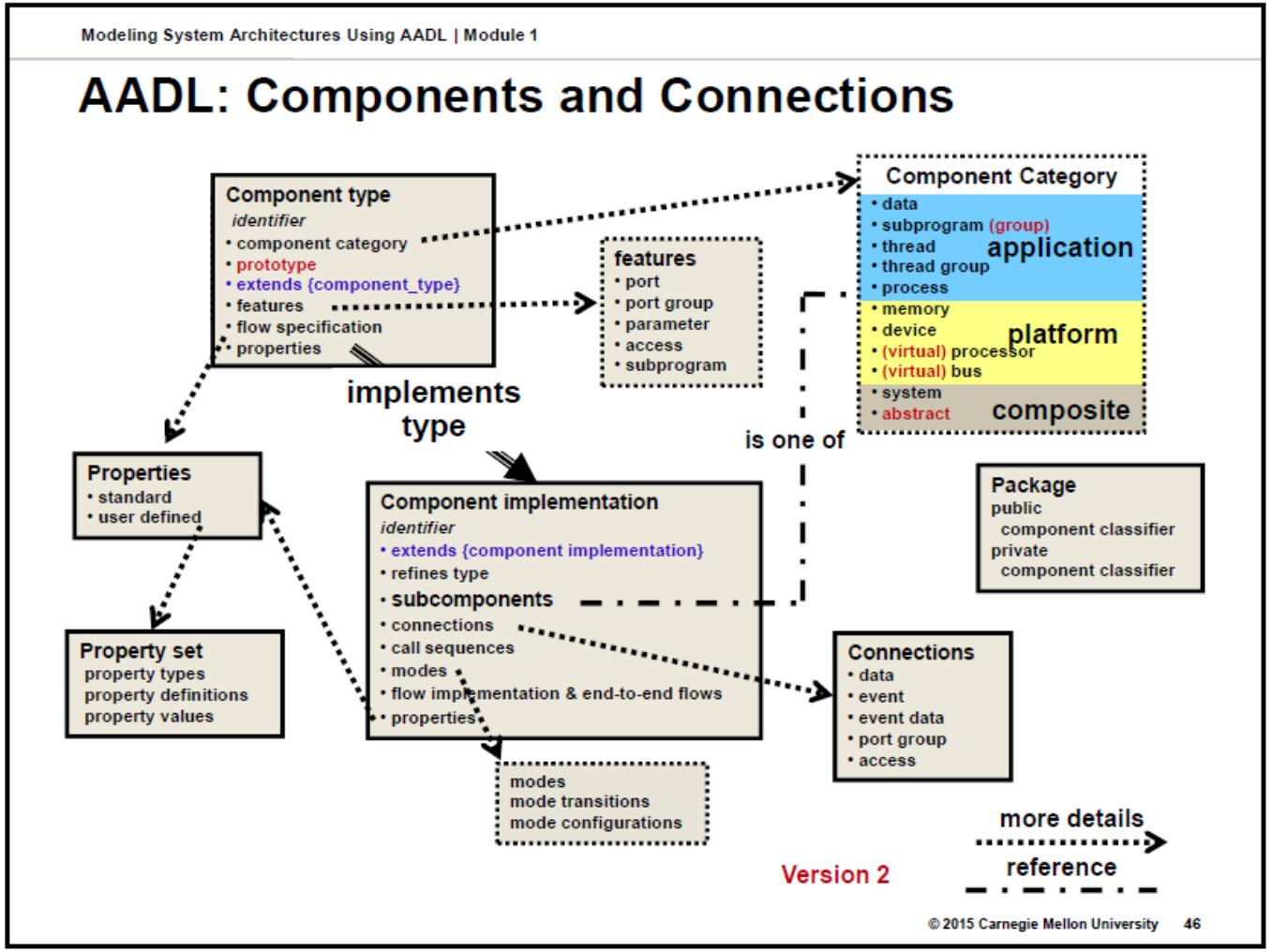
推荐
modeler.org.cn

Architecture Analysis and Design Language (AADL)

Part 2

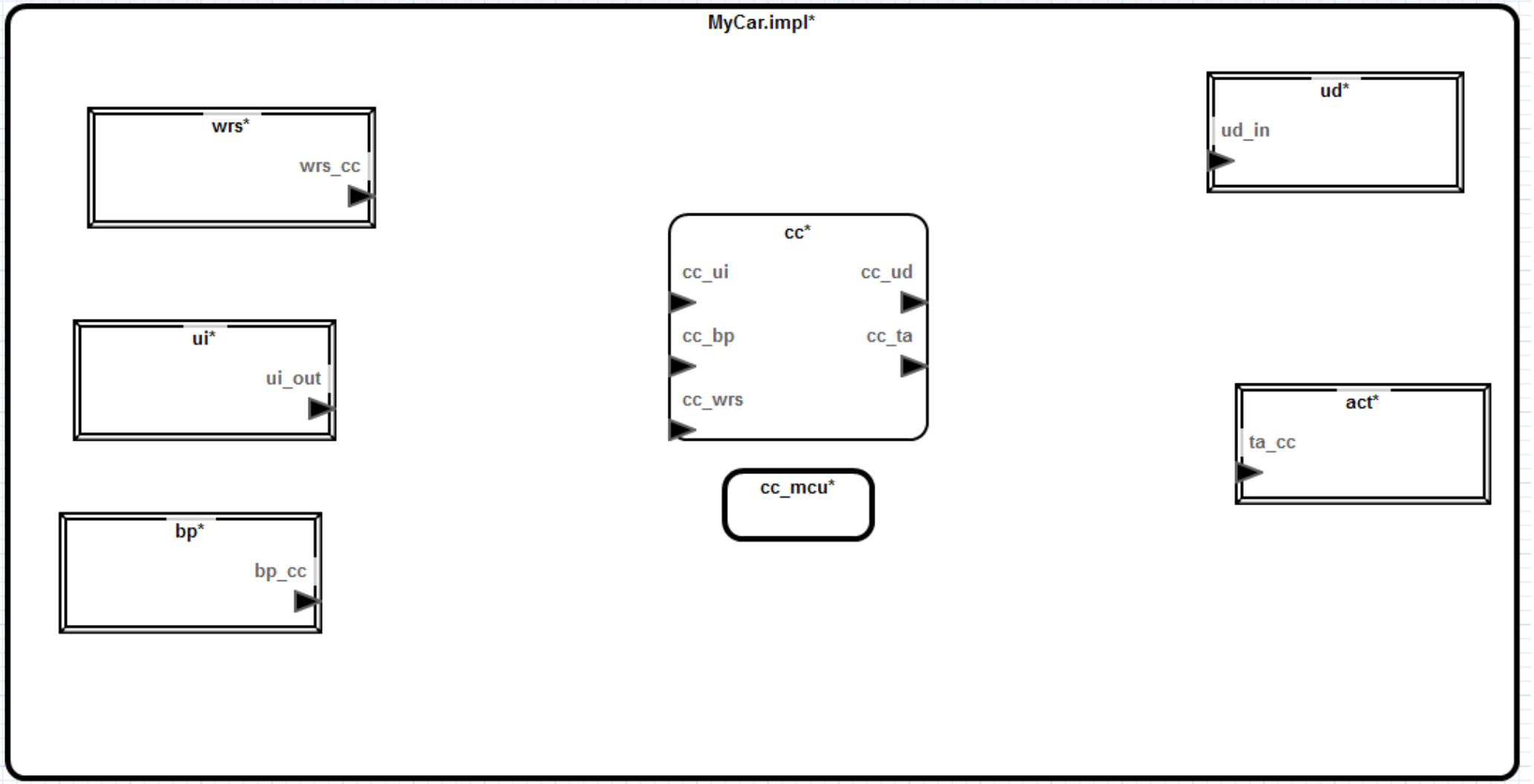
Recap

推荐
modeler.org.cn



Recap: My Car System

推荐
modeler.org.cn

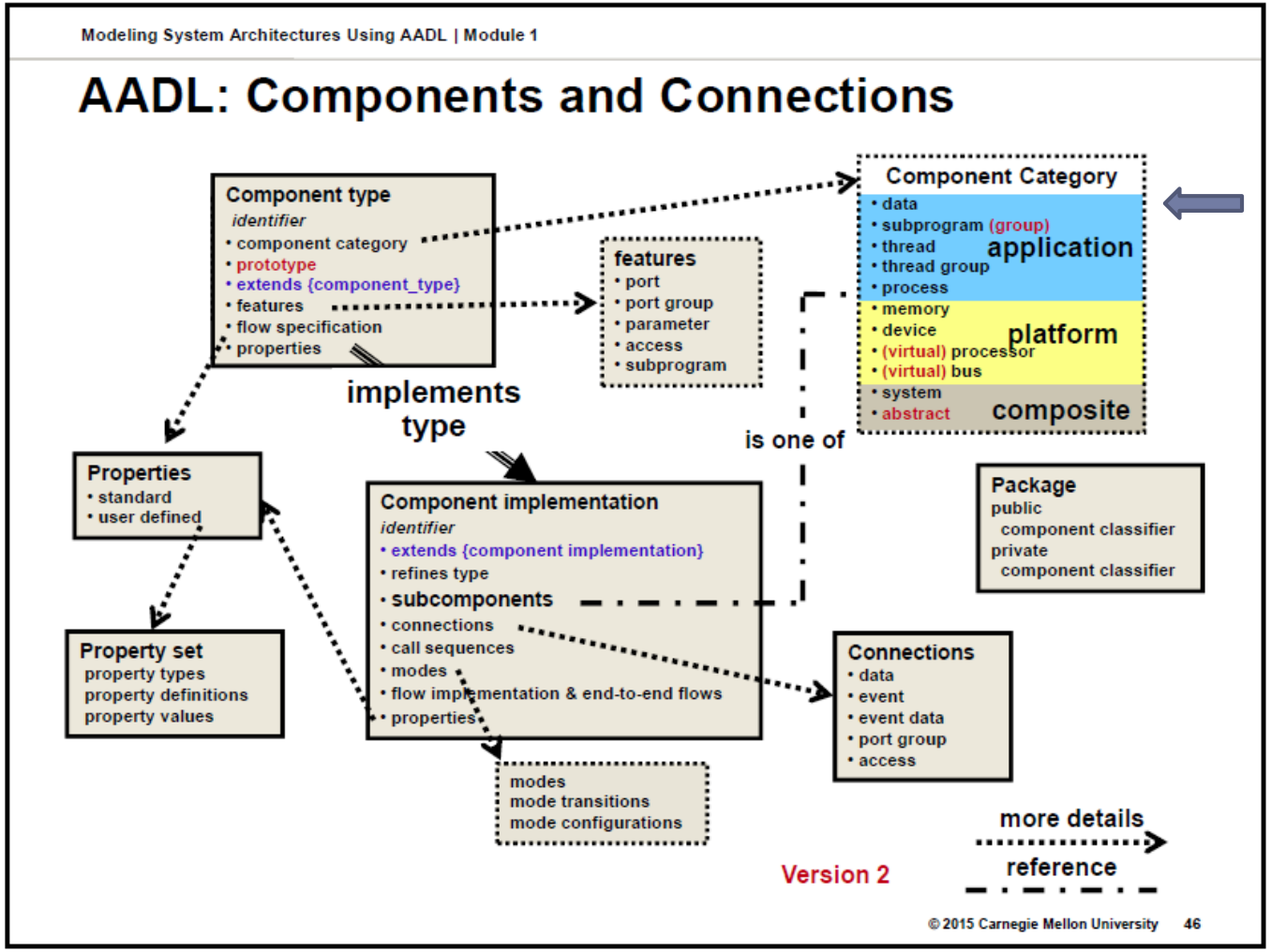


Today's topics

推荐
modeler.org.cn

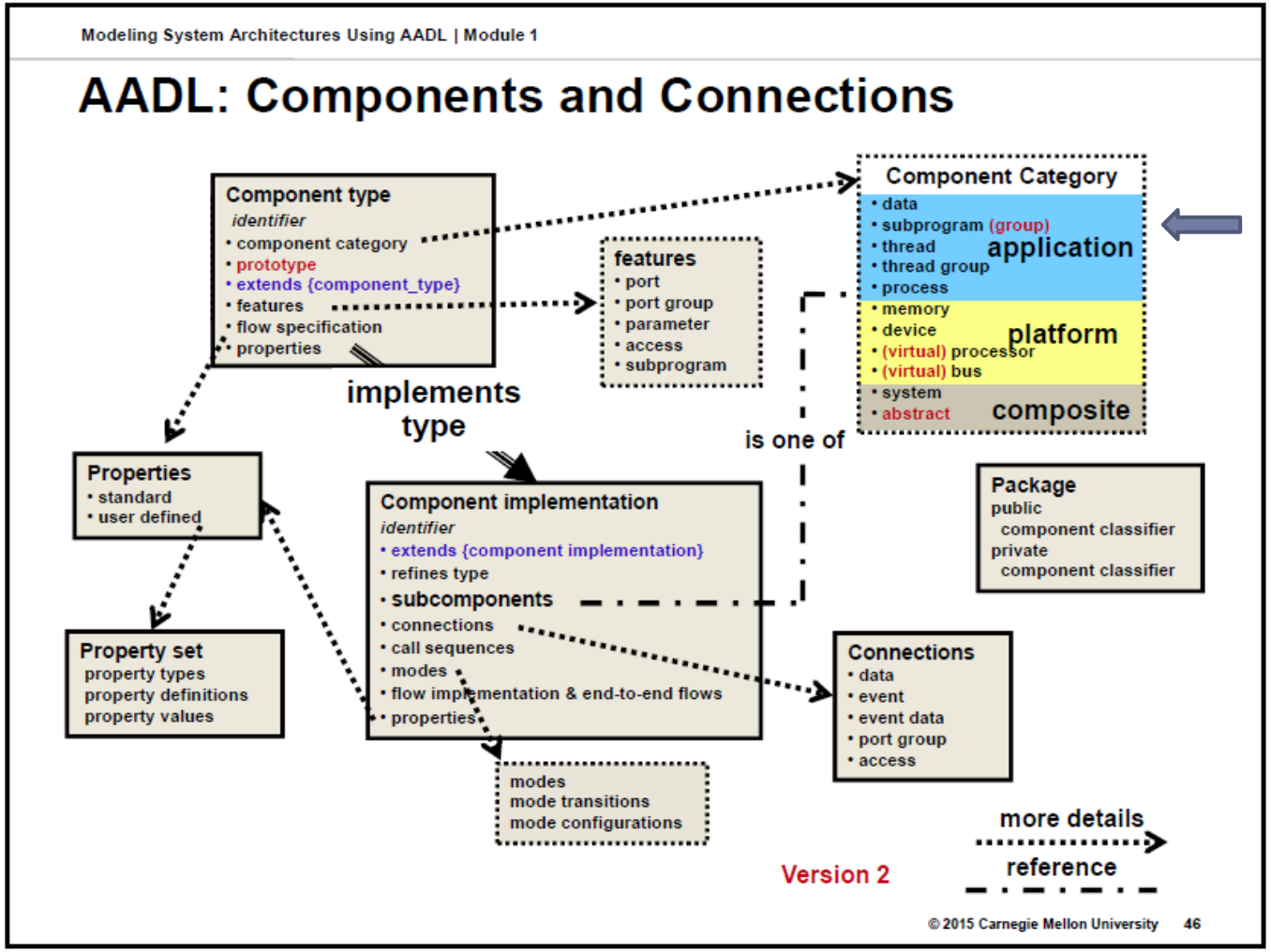
- Thread
- Process
- Connection
- Flow
- Latency analysis

Data Category



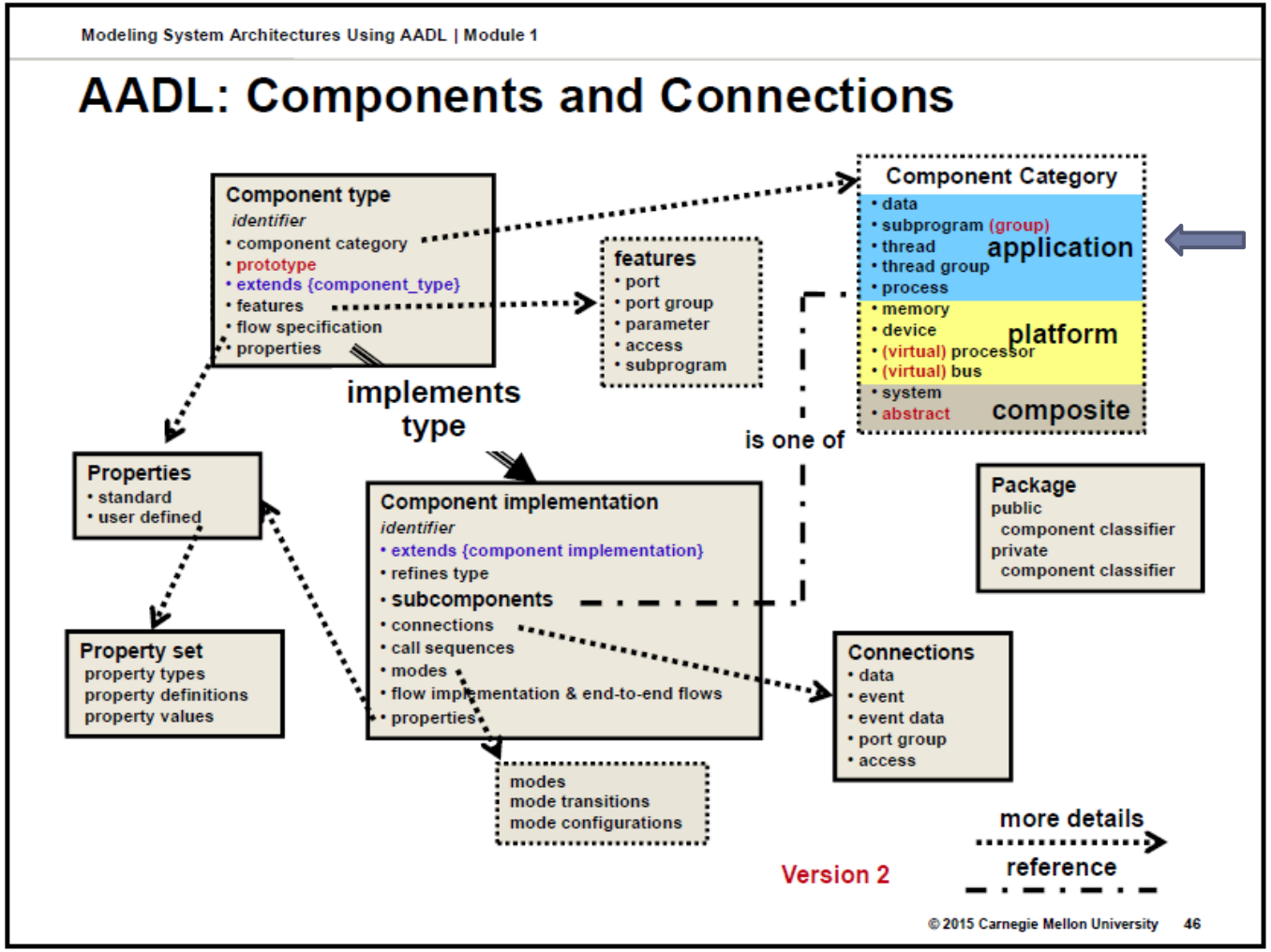
The data category captures a data type with its characteristics, such as size, representation clause, or encoding.

Subprogram Category



The subprogram category specifies a block of executable code with its interface (called parameters) that is eventually executed by a thread.

Thread Category



The thread category represents an entity that executes an instruction flow (AADL subprogram).

A thread is characterized by its timing constraints (e.g. period and deadline) and how it is executed (time or event triggered).

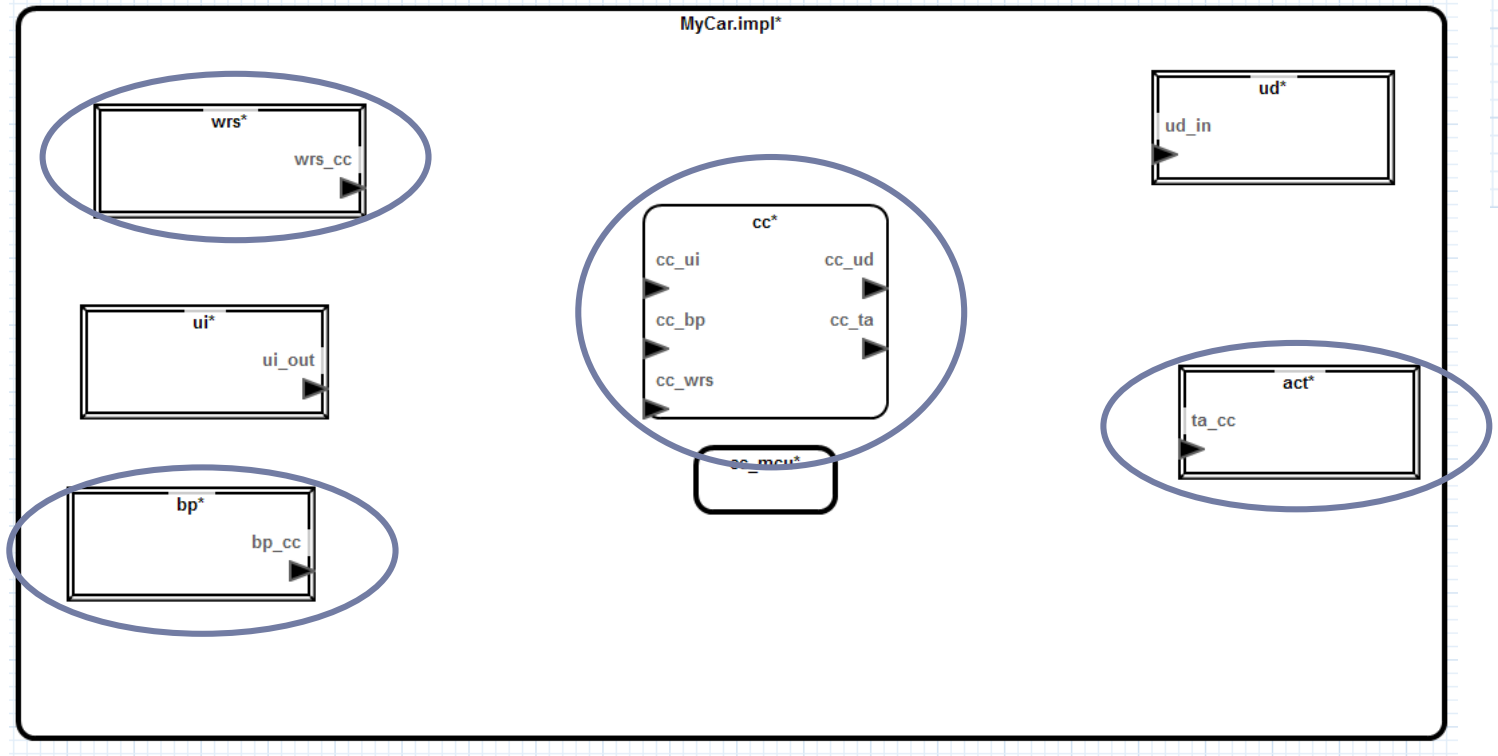
Cruise Control System: Threads

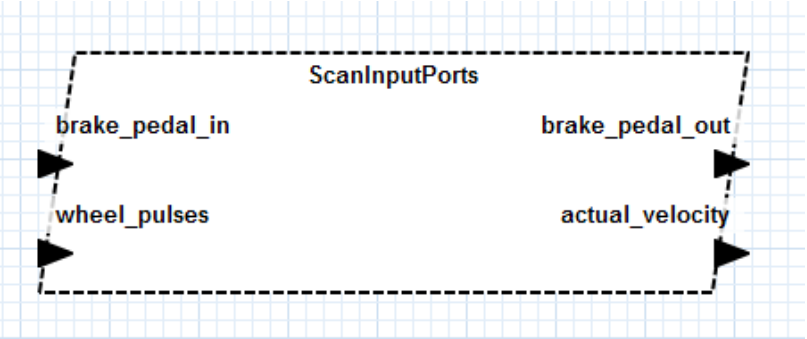
推荐
modeler.org.cn

- Acquire data from sensors and apply digital filters where required
- Control law, often a proportional-sum-difference control law
- Covert and scale the output to the throttle

Cruise Control System: Threads

- Acquire data from sensors and apply digital filters where required
- Control law, often a proportional-sum-difference control law
- Covert and scale the output to the throttle

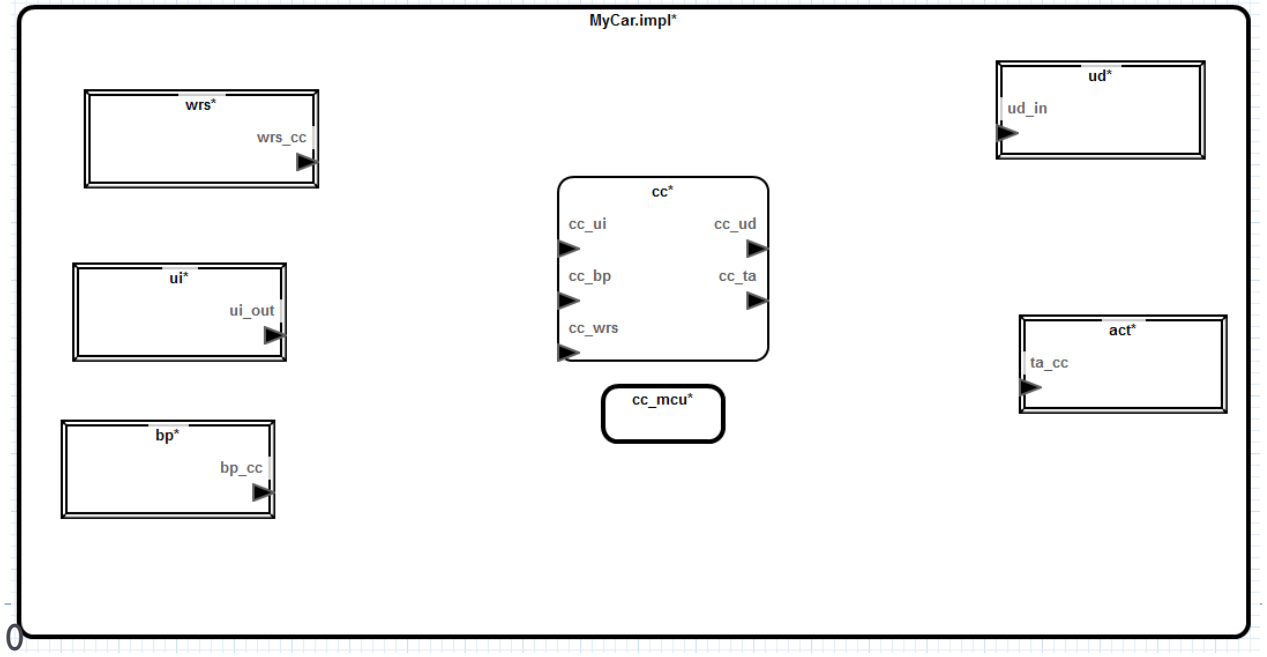


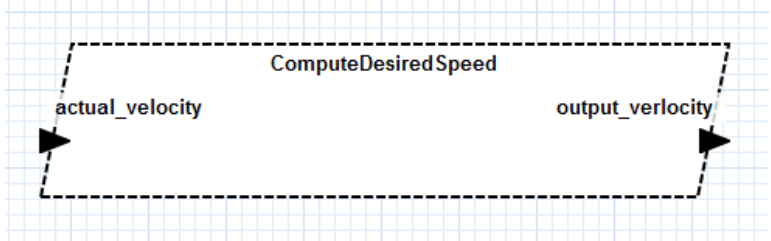


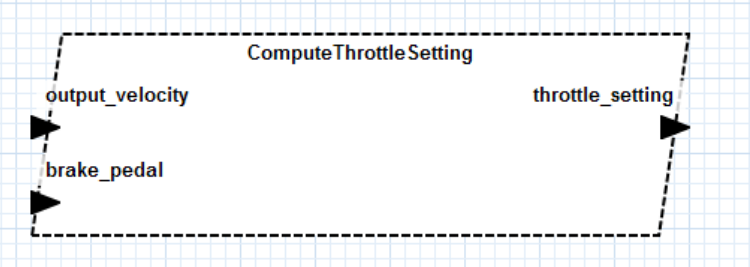

```
thread ScanInputPorts
    features
        wheel_pulses: in data port;
        brake_pedal_in: in data port;
        actual_velocity: out data port;
        brake_pedal_out: out data port;
end ScanInputPorts;
```

Cruise Control System: Threads (Exercise)

- ScanInputPorts: Acquire data from sensors and apply digital filters where required
- ComputeDesiredSpeed: Control law, often a proportional-sum-difference control law
- ComputeThrottleSetting: Covert and scale the output to the throttle

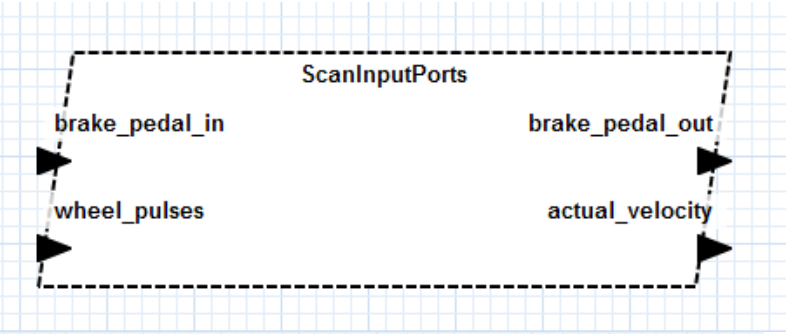


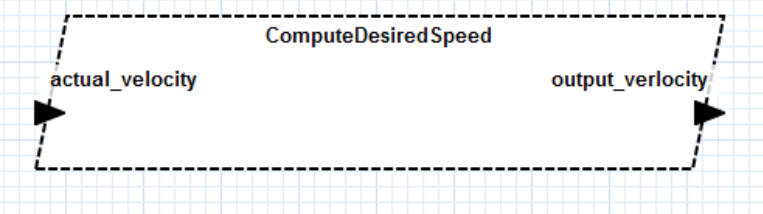


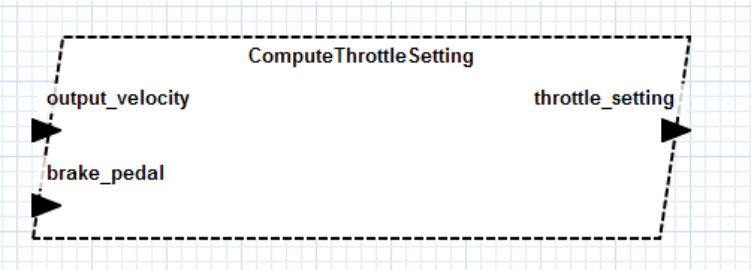


Cruise Control System: Threads

推荐
modeler.org.cn

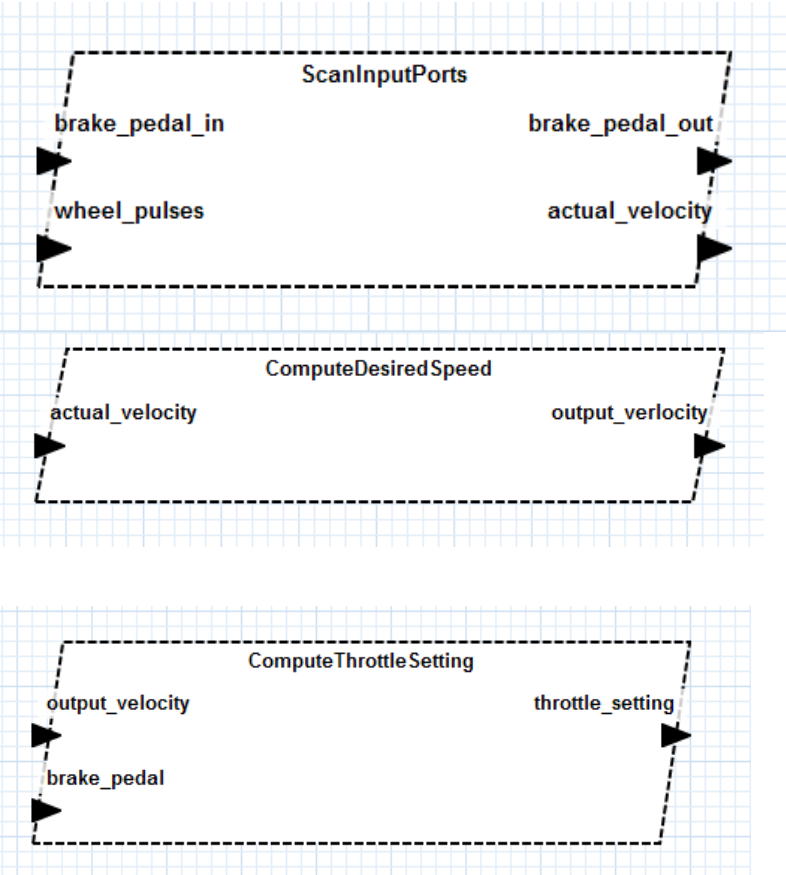







```
package CCThreads
public
    thread ScanInputPorts
        features
            wheel_pulses: in data port;
            brake_pedal_in: in data port;
            actual_velocity: out data port;
            brake_pedal_out: out data port;
    end ScanInputPorts;

    thread ComputeDesiredSpeed
        features
            actual_velocity: in data port;
            output_verlocity: out data port;
    end ComputeDesiredSpeed;

    thread ComputeThrottleSetting
        features
            output_velocity: in data port;
            brake_pedal: in data port;
            throttle_setting: out data port;
    end ComputeThrottleSetting;

end CCThreads;
```

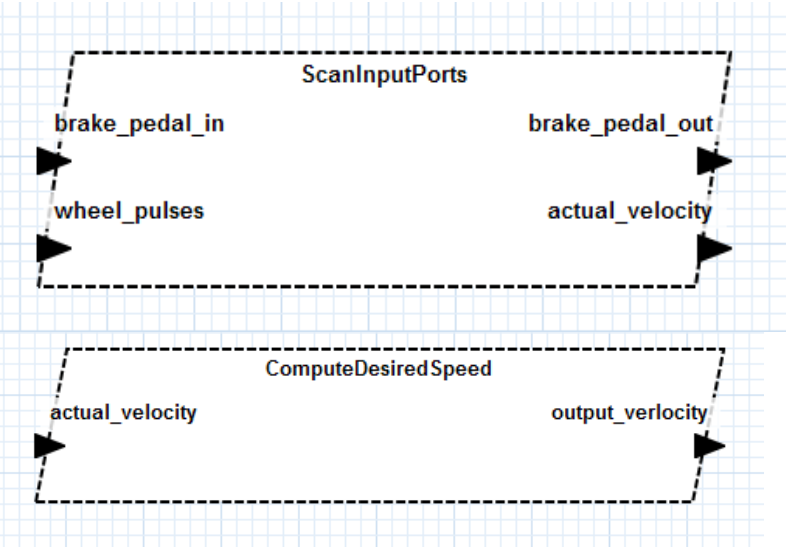
Cruise Control System: Threads and Properties (Exercise

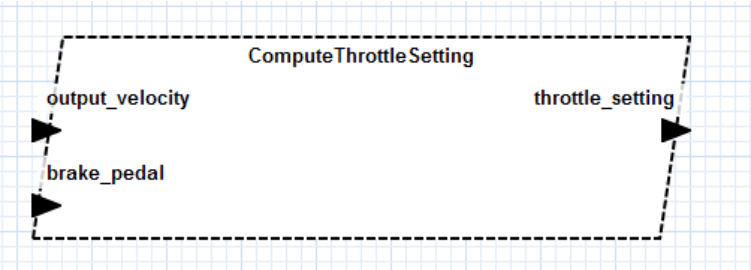


- ScanInputPorts
  - Dispatch_protocol = periodic
  - Period = 5ms
  - Compute_Execution_time = 3ms
- ComputeDesiredSpeed
  - Dispatch_protocol = periodic
  - Period = 5ms
  - Compute_Execution_time = 3ms
- ComputeThrottleSetting
  - Dispatch_protocol = periodic
  - Period = 5ms
  - Compute_Execution_time = 3ms

Cruise Control System: Threads and Properties

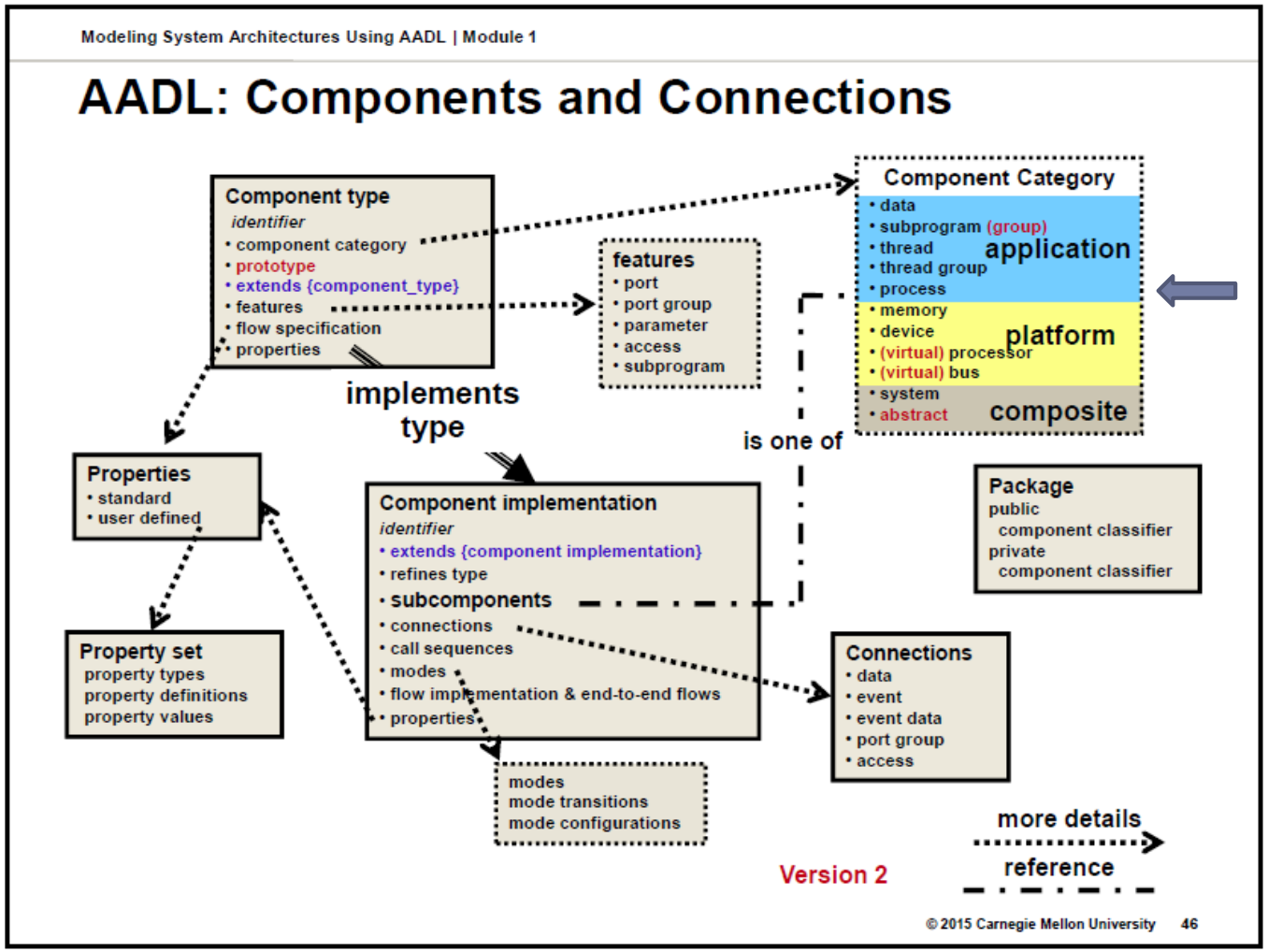
推荐
modeler.org.cn






```
package CCThreads
public
    thread ScanInputPorts
        features
            wheel_pulses: in data port;
            brake_pedal_in: in data port;
            actual_velocity: out data port;
            brake_pedal_out: out data port;
        properties
            Dispatch_Protocol => Periodic;
            Period => 5Ms;
            Compute_Execution_Time => 3Ms .. 3Ms;
    end ScanInputPorts;

    thread ComputeDesiredSpeed
        features
            actual_velocity: in data port;
            output_verlocity: out data port;
        properties
            Dispatch_Protocol => Periodic;
            Period => 10Ms;
            Compute_Execution_Time => 2Ms .. 2Ms;
    end ComputeDesiredSpeed;

    thread ComputeThrottleSetting
        features
            output_velocity: in data port;
            brake_pedal: in data port;
            throttle_setting: out data port;
        properties
            Dispatch_Protocol => Periodic;
            Period => 5Ms;
            Compute_Execution_Time => 3Ms .. 3Ms;
    end ComputeThrottleSetting;
end CCThreads;
```

Process Category

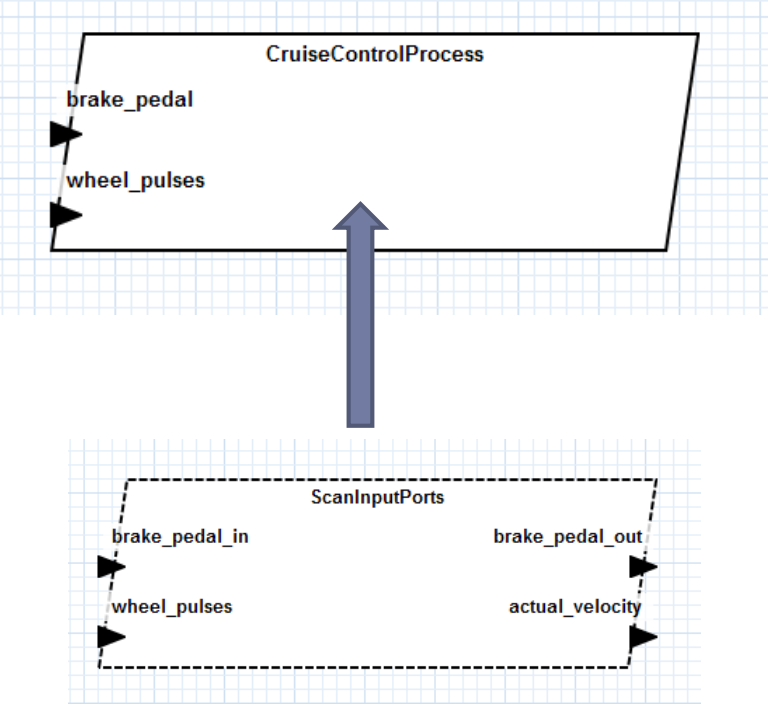


The process category represents an address space that contains data and thread subcomponents containing the necessary elements to execute the software.

Process is associated to a processor (that executes its thread and provides basic execution services) and a memory (that physically stores the memory).

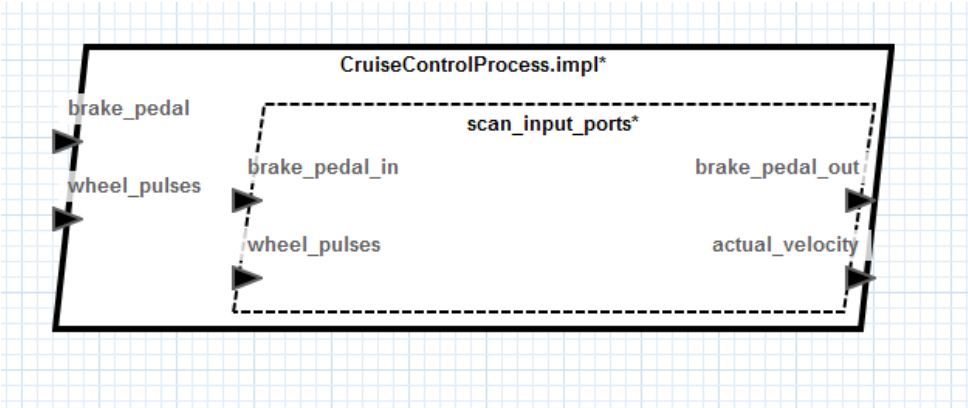
Process is an empty address space whose boundaries are enforced by the underlying runtime.

Cruise Control System: Process Type



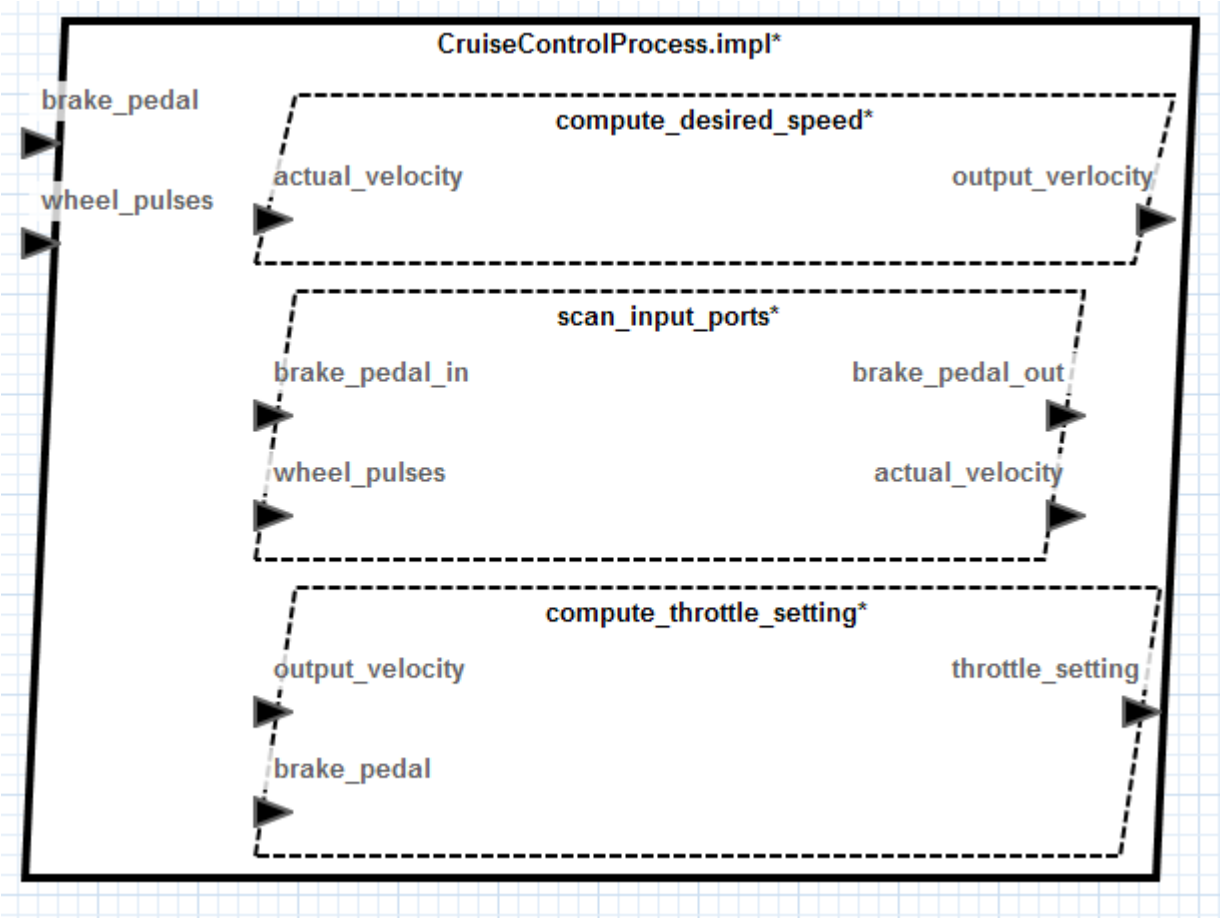

```
process CruiseControlProcess
    features
        wheel_pulses: in data port;
        brake_pedal: in data port;
end CruiseControlProcess;
```

Cruise Control System: Process Implementation

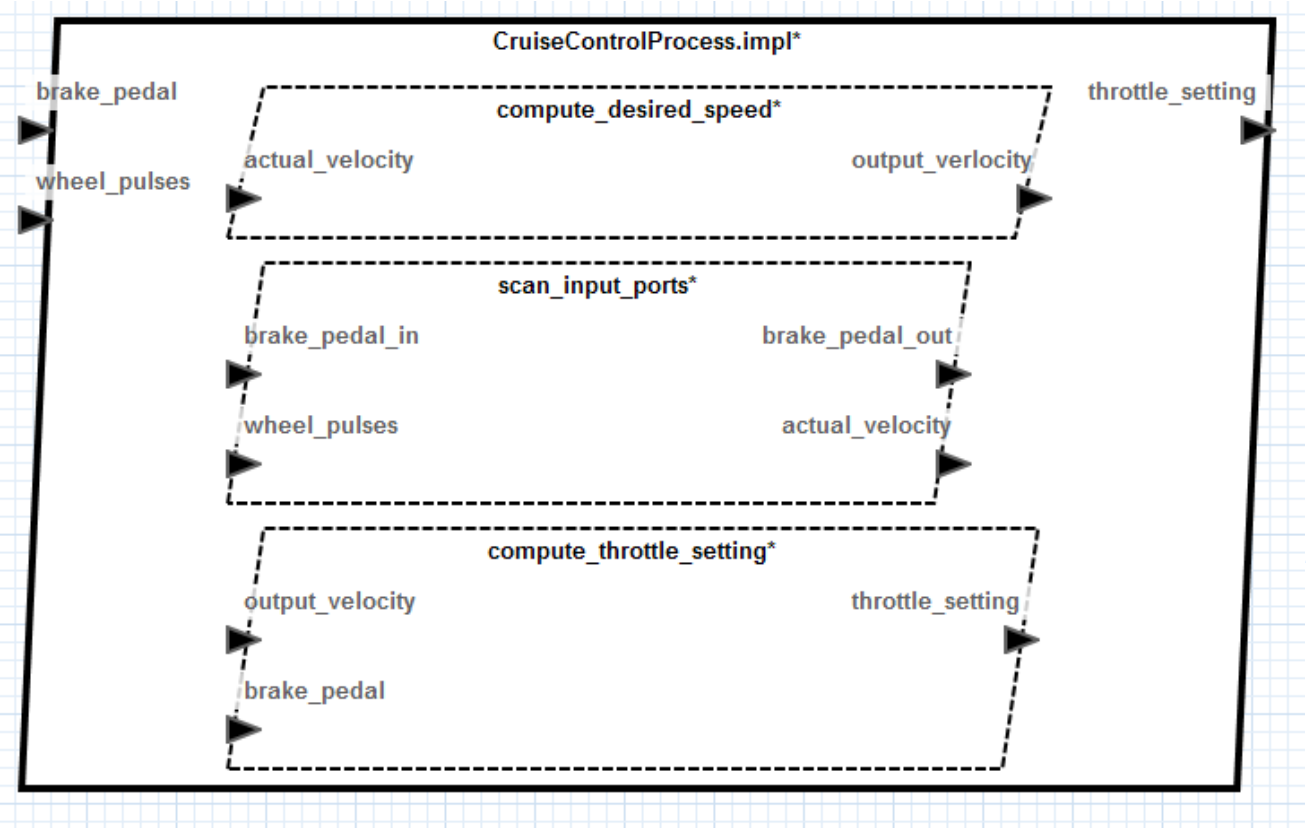



```
process CruiseControlProcess
    features
        wheel_pulses: in data port;
        brake_pedal: in data port;
end CruiseControlProcess;

process implementation CruiseControlProcess.impl
subcomponents
  scan_input_ports: thread ScanInputPorts;
end CruiseControlProcess.impl;
```

Cruise Control System: Process (Exercise)

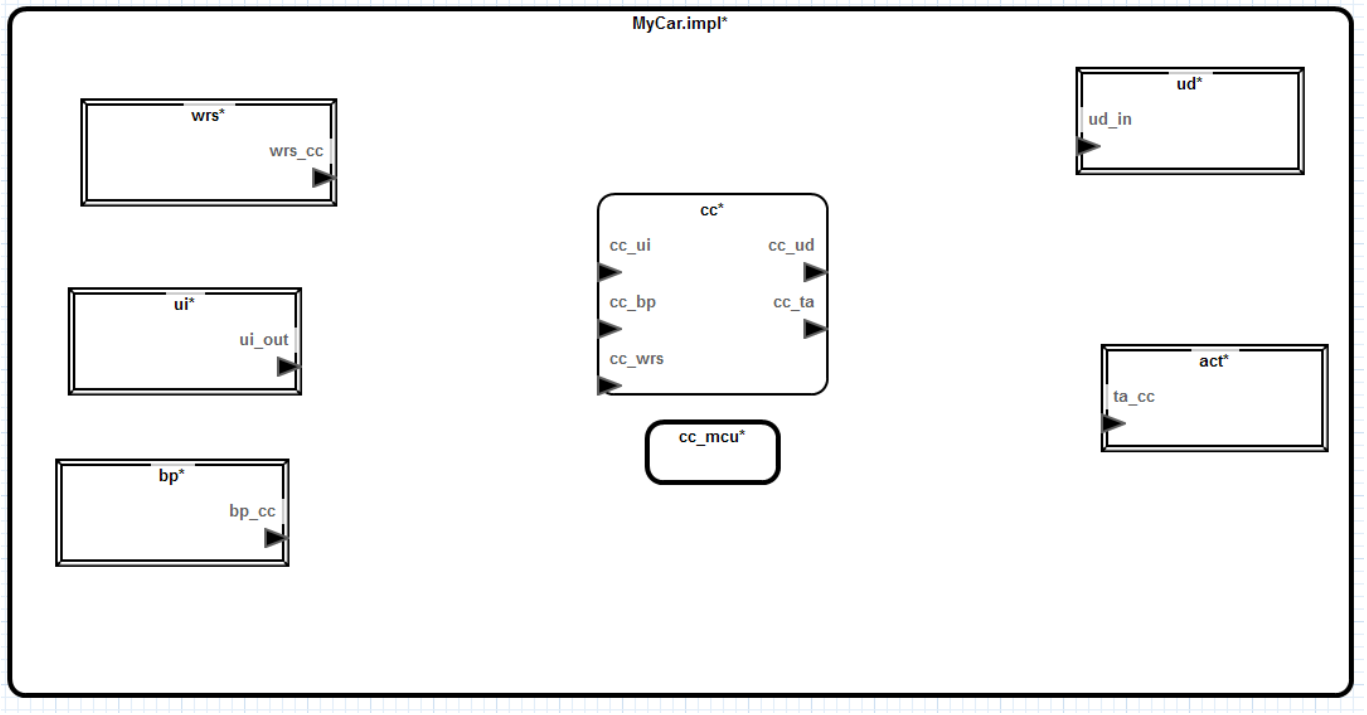


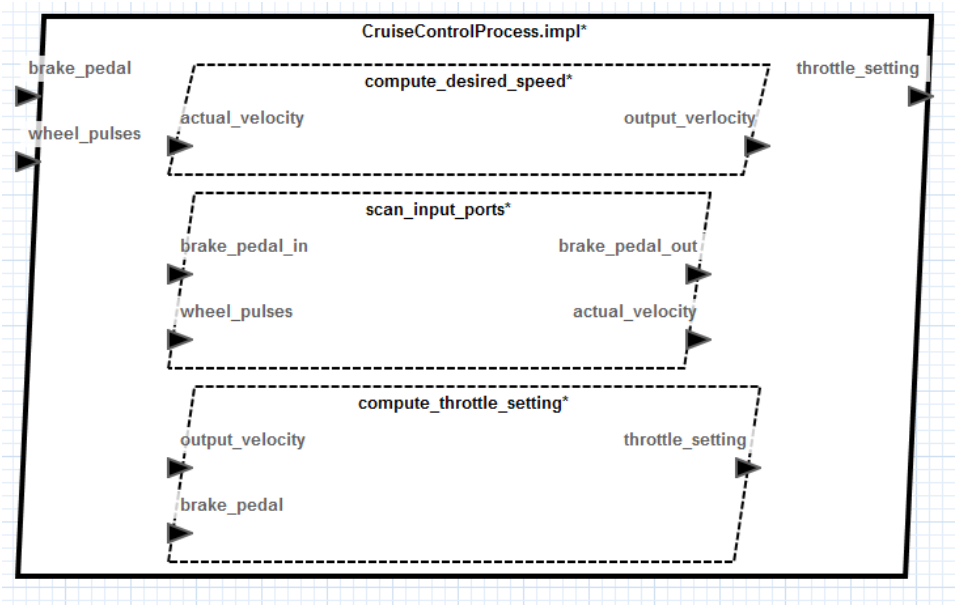
Cruise Control System: Process Implementation



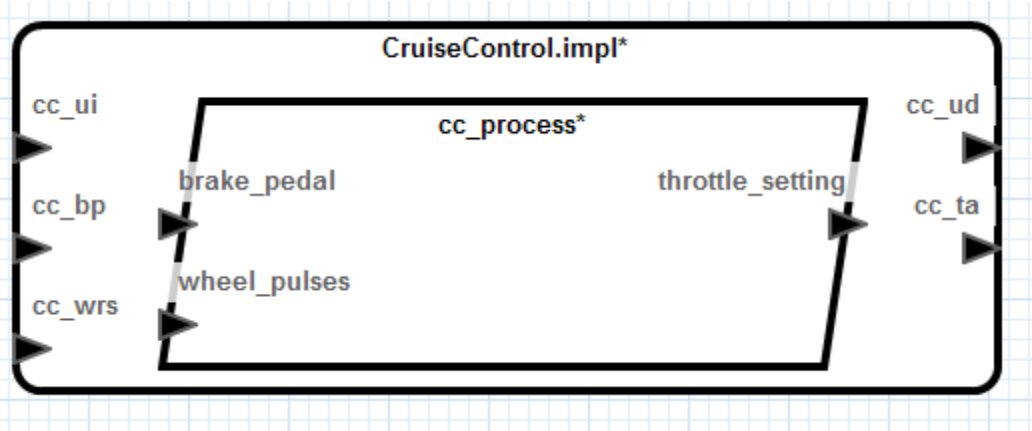

```
process CruiseControlProcess
    features
        wheel_pulses: in data port;
        brake_pedal: in data port;
        throttle_setting: out data port;
end CruiseControlProcess;

process implementation CruiseControlProcess.impl
    subcomponents
        scan_input_ports: thread ScanInputPorts;
        compute_desired_speed: thread ComputeDesiredSpeed;
        compute_throttle_setting: thread ComputeThrottleSetting;
end CruiseControlProcess.impl;
```

Cruise Control System: System and Process (Exerc





Cruise Control System: System and Process

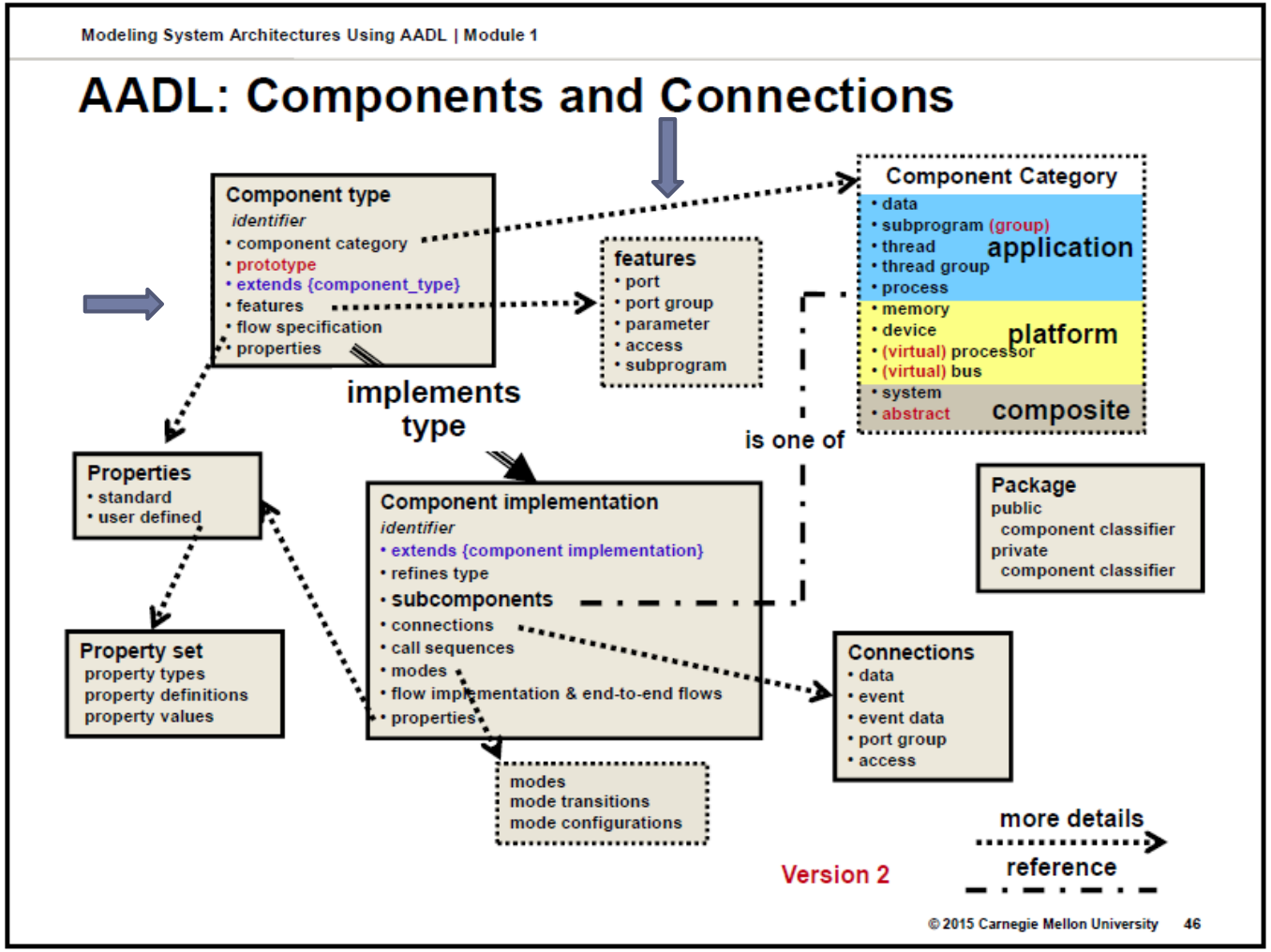



```
system CruiseControl
    features
        cc_ui: in data port;
        cc_ud: out data port;
        cc_wrs: in data port;
        cc_bp: in data port;
        cc_ta: out data port;
end CruiseControl;

system implementation CruiseControl.impl
    subcomponents
        cc_process: process CCThreads::CruiseControlProcess.impl;
end CruiseControl.impl;
```

Component Type: Features

Feature defines the component interfaces available to the outside world.

AADL interfaces show what the component exposes to the outside world, either physically (hardware component) or logically (software component).



Port represents an interaction point for an event, a data transfer, or a combination of both.

3 types of ports:

a. Event port
b. Data port
c. Event data port

Access specifies if a component requires or provides access to a particular service. E.g. a processor requires access to an Ethernet bus.

2 types of accesses:

a. Bus access
b. Data access.

Ports are characterized by a direction: in, out, or in out.

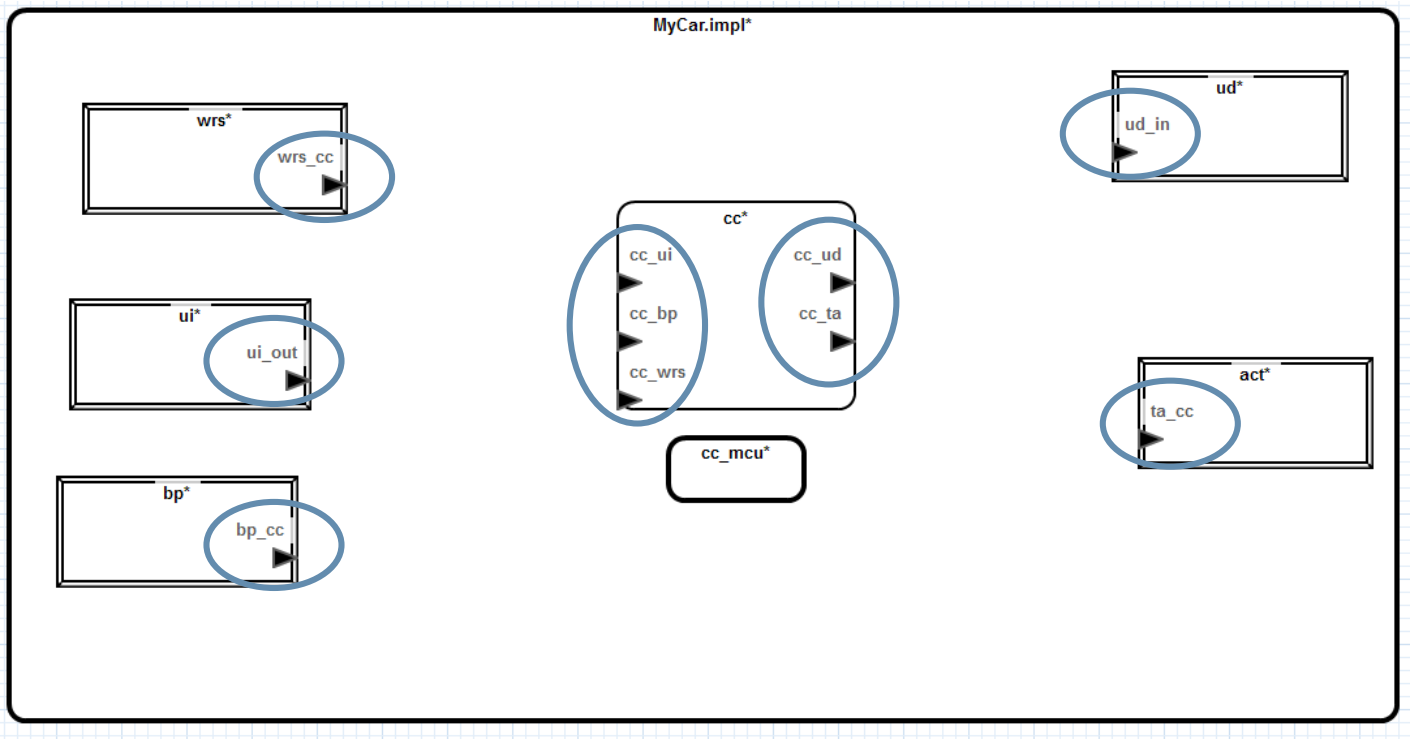
Ports

推荐
modeler.org.cn

- Event port
  - Event port represents an interface for emission/reception of a notification and carrying a signal without any data.
  - Example:
    - Health monitoring exception
    - Detection of data from a sensor when the temperature below a predefined threshold
- Data port
  - Data port represents an interface that receives a data flow and always keeps the latest values without queuing.
  - The port value is updated as soon as a new data instance is received.
- Event data port
  - Event data port represents interfaces for transporting a signal/event associated with data with queuing.

Component Type: Ports

推荐
modeler.org.cn




```
device WheelRotationSensor
    features
        wrs_cc: out data port;
    properties
        SEI::NetWeight => 0.25kg;
end WheelRotationSensor;
```

```
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;
end CruiseControl;
```

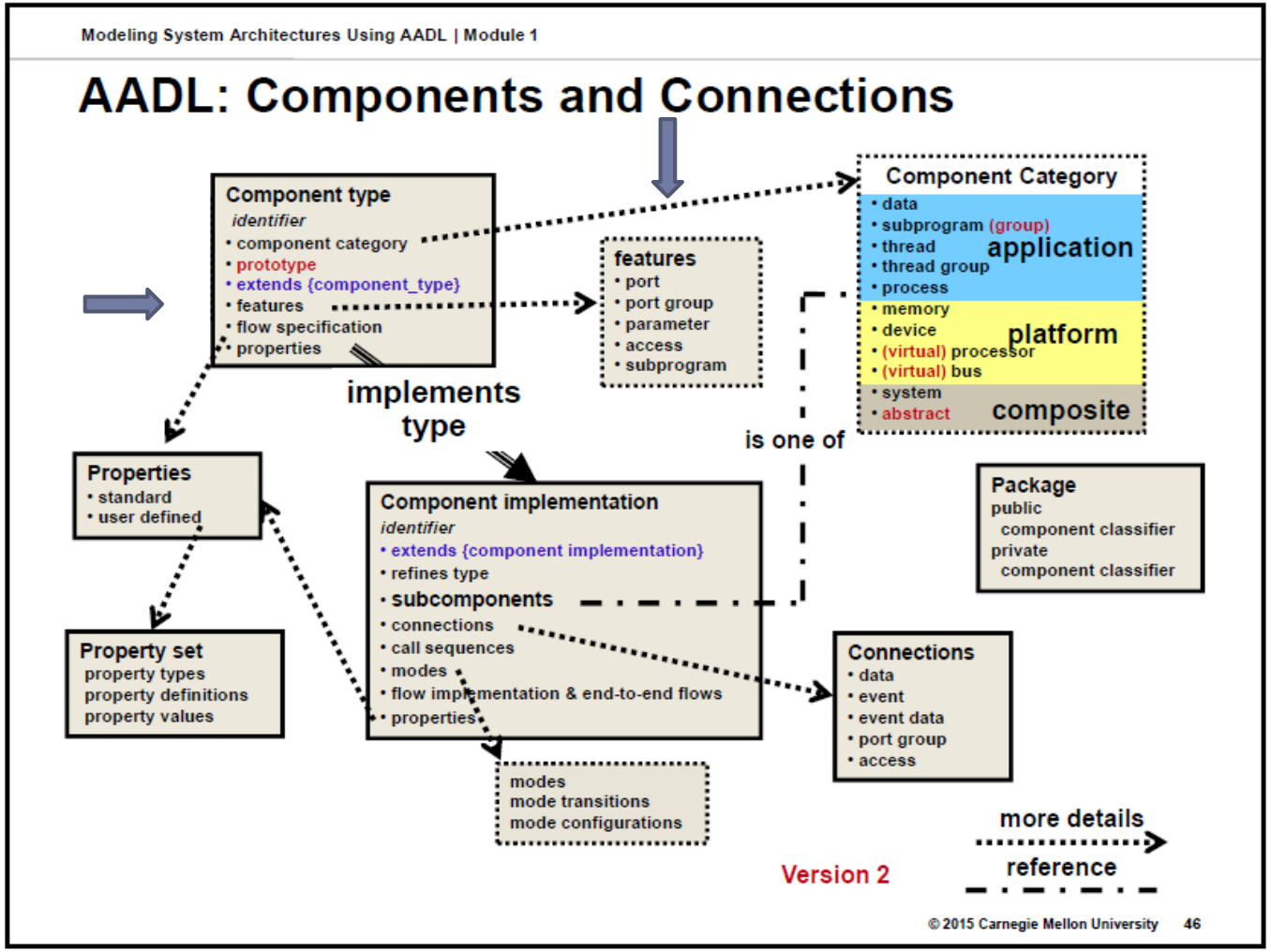
```
device ThrottleActuator
    features
        ta_cc: in data port;
    properties
        SEI::NetWeight => 2.0kg;
end ThrottleActuator;
```

Component Implementation: Connections

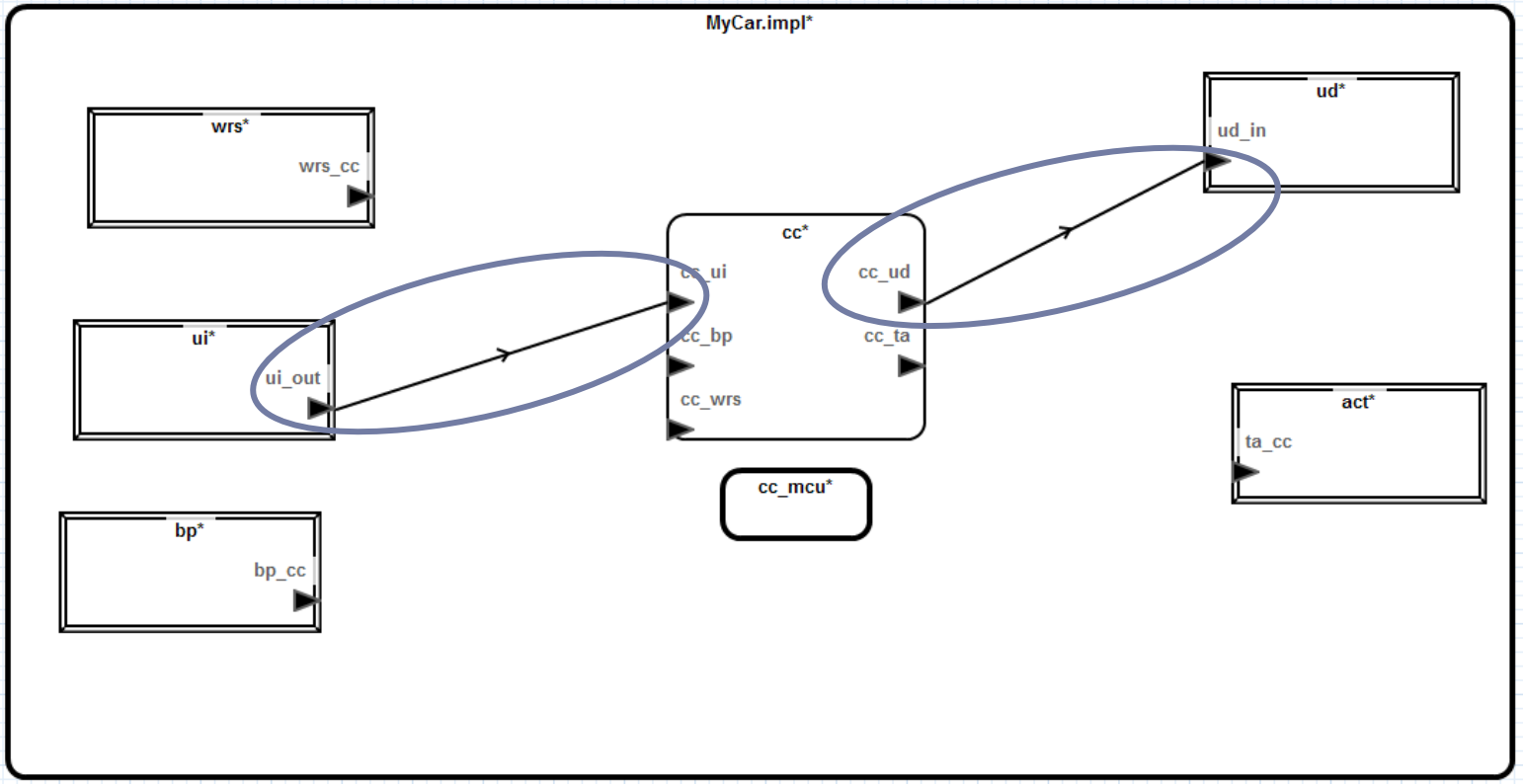
Feature defines the component interfaces available to the outside world.

Connections define how the component features (defined in the type) are connected to the subcomponent features.

Connections show how interfaces from the outside world interact with the interfaces located within the component.

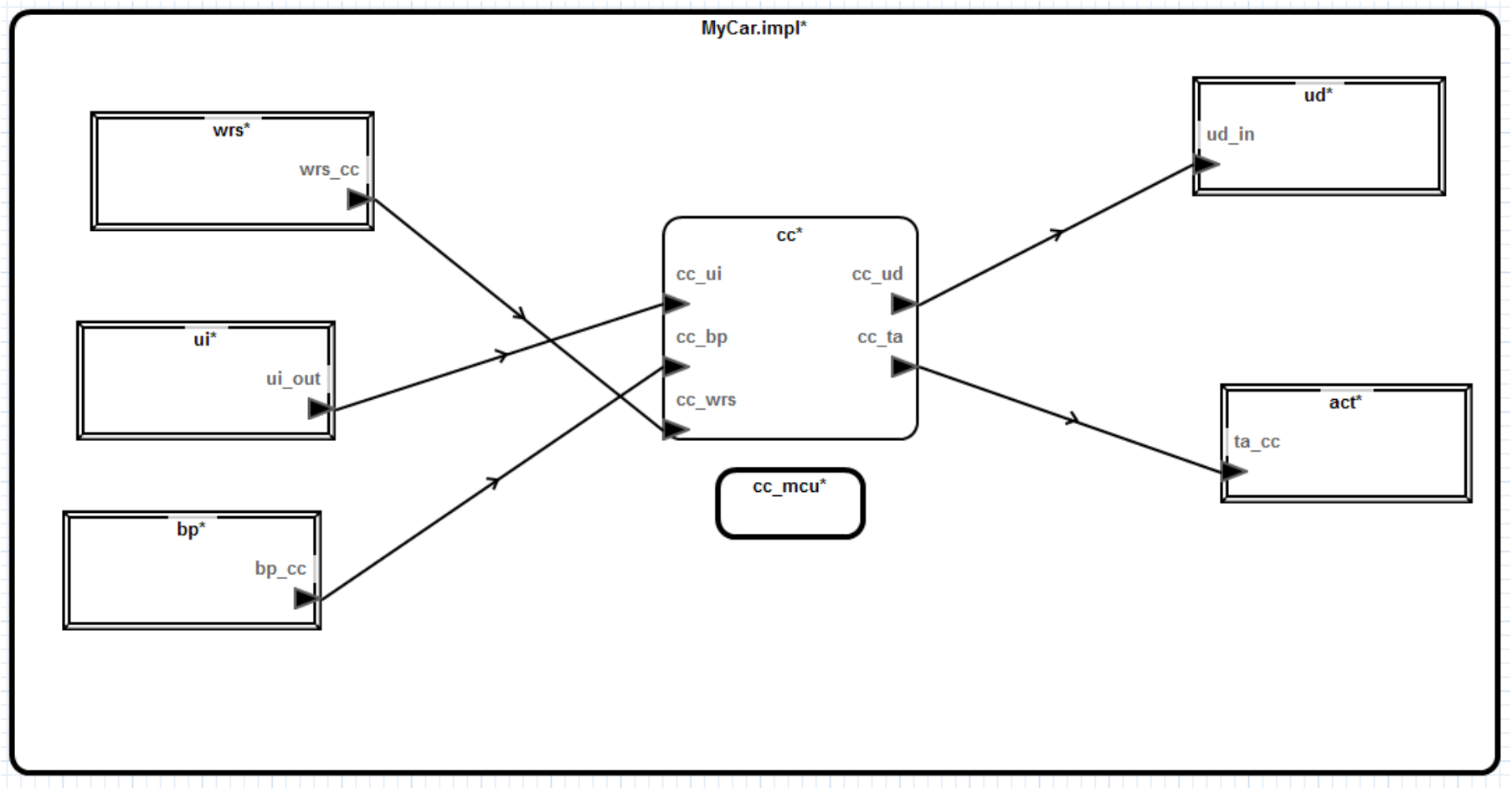


Component Implementation: Connections

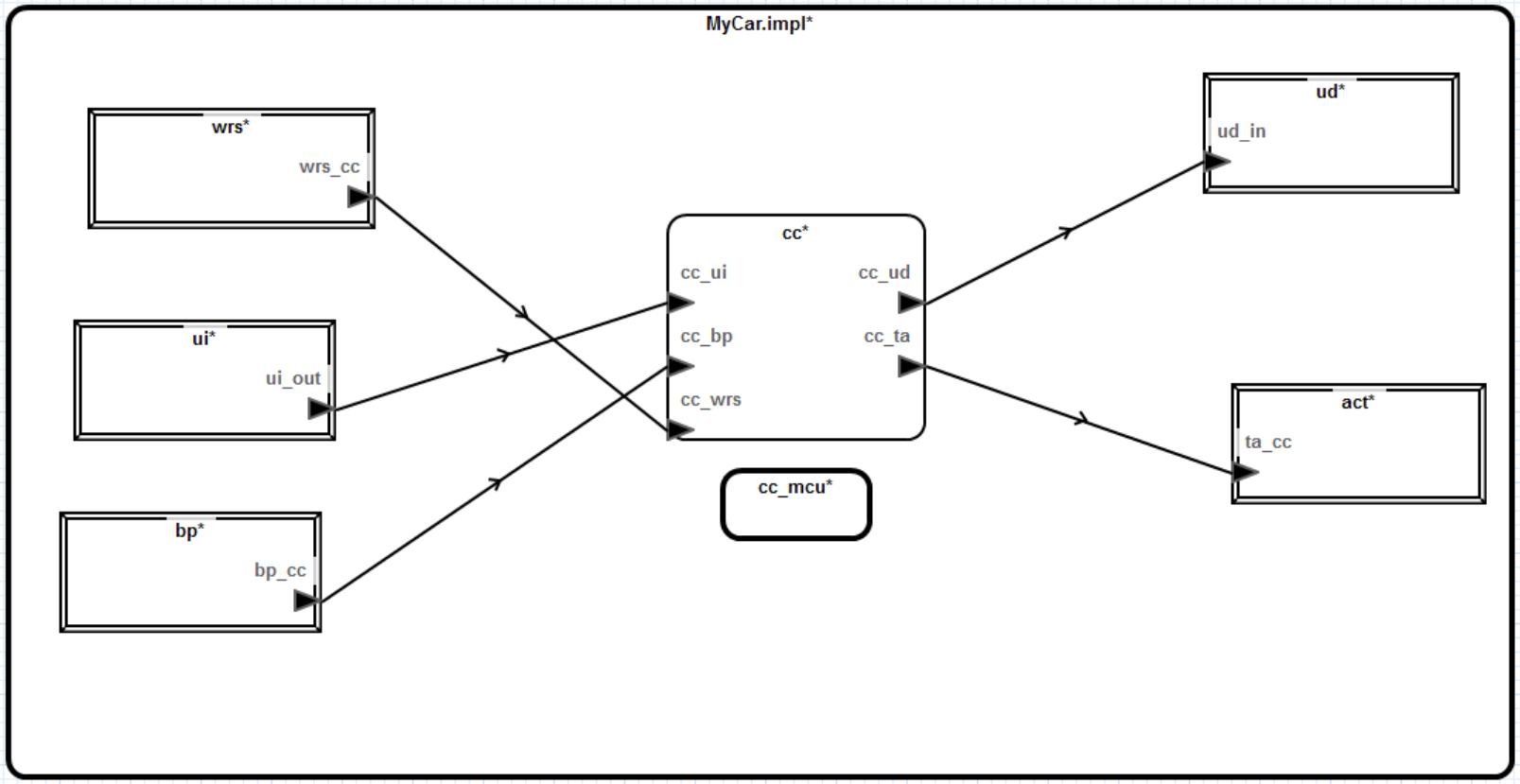



```
system implementation MyCar.impl
    subcomponents
        ui: device UserInterface::UserInput;
        ud: device UserInterface::UserDisplay;
        cc: system SoftwareApps::CruiseControl;
        wrs: device Sensors::WheelRotationSensor;
        bp: device Sensors::BrakePedal;
        act: device Actuators::ThrottleActuator;
        cc_mcu: system HardwarePlatform::cc_mcu.impl;
    connections
        c1: port ui.ui_out -> cc.cc_ui;
        c2: port cc.cc_ud -> ud.ud_in;
end MyCar.impl;
```

Component Implementation: Connections (Exercise



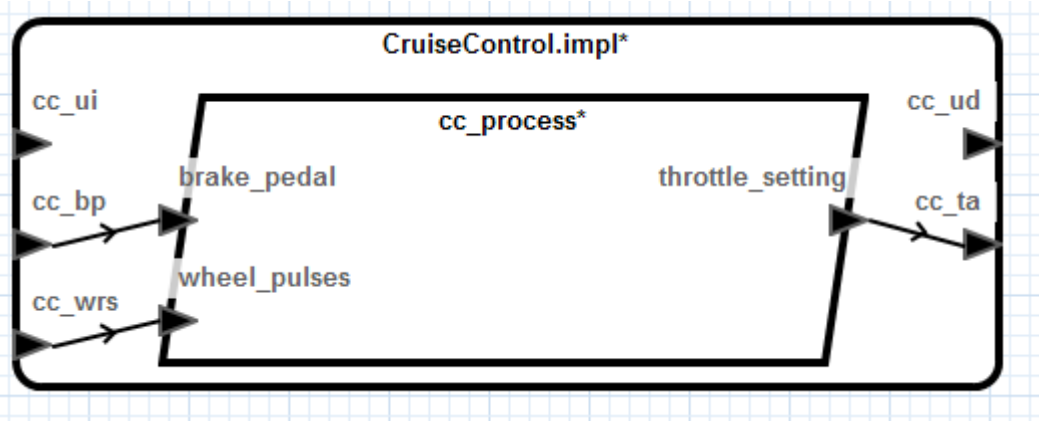
Component Implementation: Connections



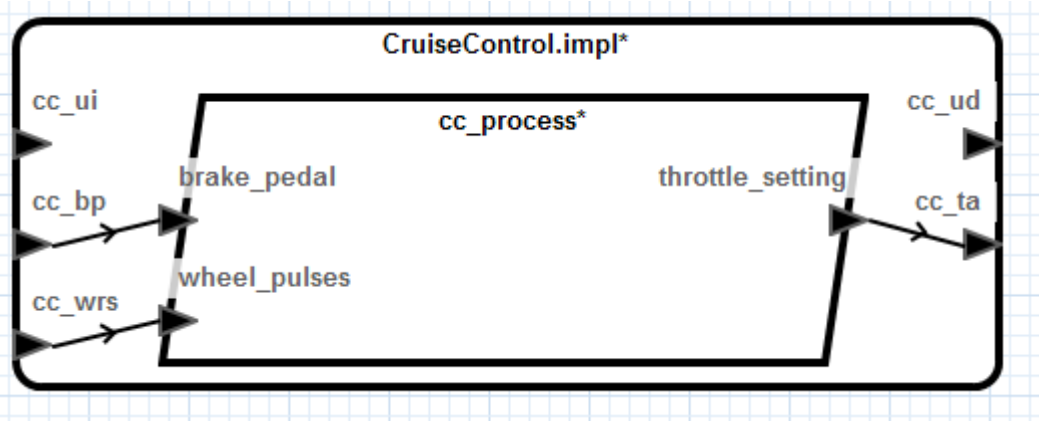

```
system implementation MyCar.impl
    subcomponents
        ui: device UserInterface::UserInput;
        ud: device UserInterface::UserDisplay;
        cc: system SoftwareApps::CruiseControl;
        wrs: device Sensors::WheelRotationSensor;
        bp: device Sensors::BrakePedal;
        act: device Actuators::ThrottleActuator;
        cc_mcu: system HardwarePlatform::cc_mcu.impl;
    connections
        c1: port ui.ui_out -> cc.cc_ui;
        c2: port cc.cc_ud -> ud.ud_in;
        c3: port wrs.wrs_cc -> cc.cc_wrs;
        c4: port bp.bp_cc -> cc.cc_bp;
        c5: port cc.cc_ta -> act.ta_cc;
end MyCar.impl;
```

Component Implementation: Connections (Exercise

推荐
modeler.org.cn



Component Implementation: Connections




```
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;
end CruiseControl;

system implementation CruiseControl.impl
    subcomponents
        cc_process: process CCThreads::CruiseControlProcess.impl;
    connections
        CruiseControl_impl_new_connection: port cc_bp -> cc_process.brake_pedal;
        CruiseControl_impl_new_connection2: port cc_wrs -> cc_process.wheel_pulses;
        CruiseControl_impl_new_connection3: port cc_process.throttle_setting -> cc_ta;
end CruiseControl.impl;
```

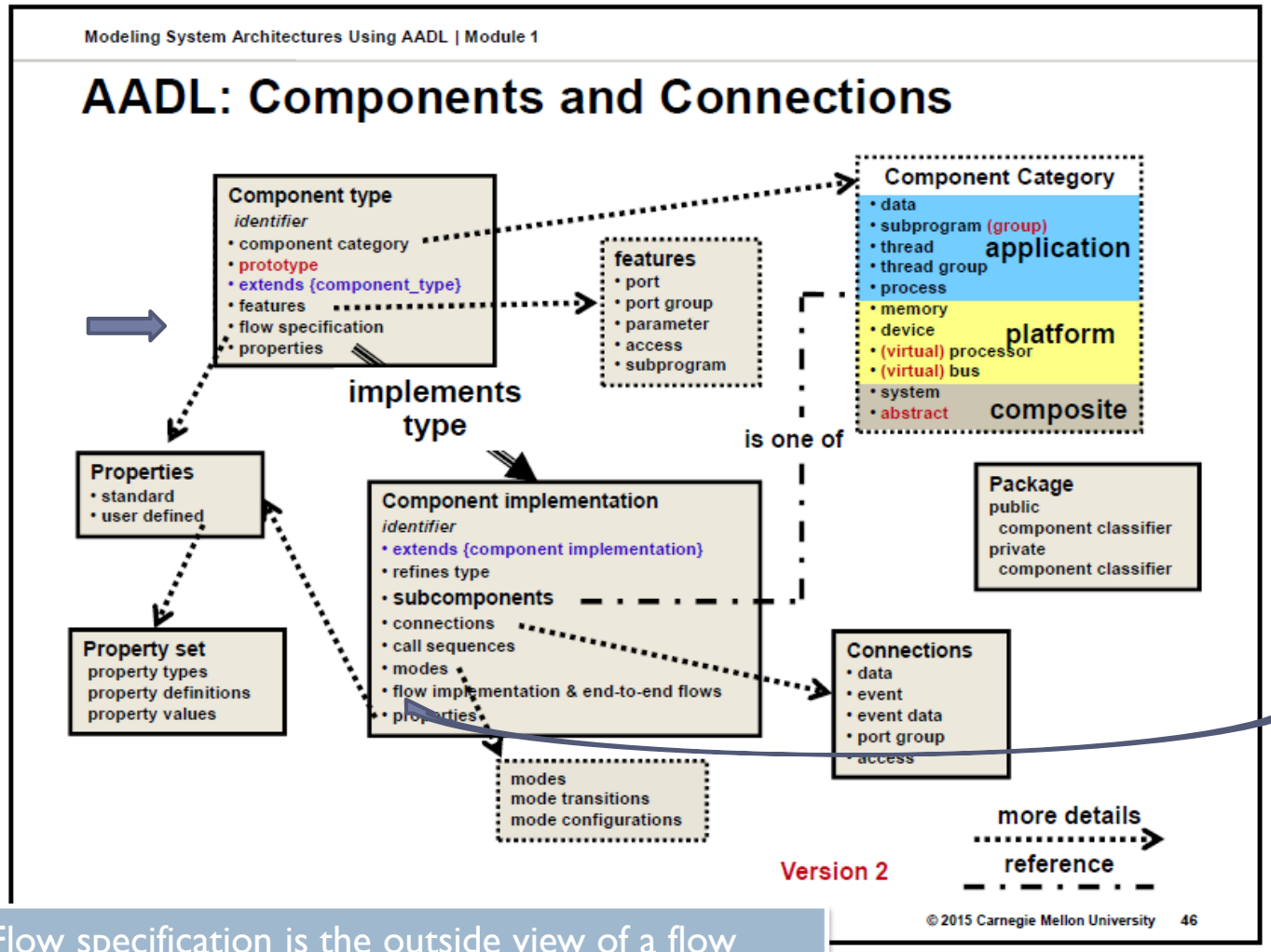
Component Type and Implementation: Flows

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 specification is the outside view of a flow through a component. It specifies the externally visible flow source, path, and sink.

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.

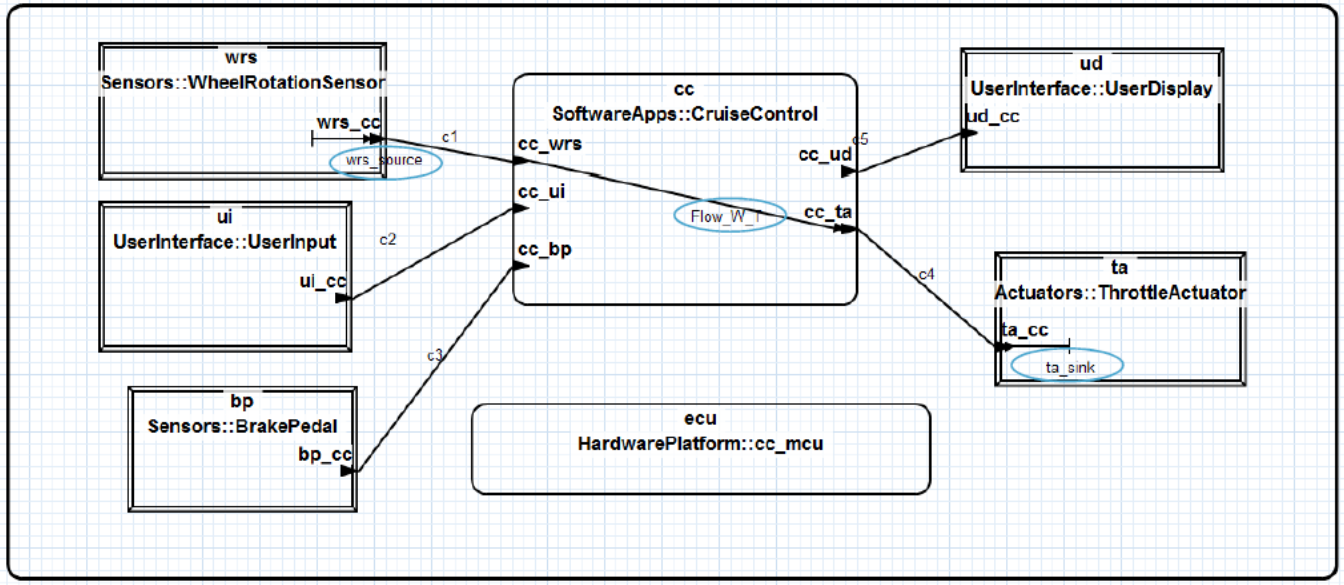
Component Type: Flow Specification

- Flow source
  - Specifies the origination of the flow
- Flow sink
  - Specifies the termination of the flow
- Flow path
  - Specifies the traversal of a component

```
device WheelRotationSensor
    features
        wrs_cc: out data port;
    flows -- Flow section goes after features.  We are defining the port as the flow source.
        wrs_source: flow source wrs_cc{
            Latency => 5ms..5ms;
        };
    properties
        SEI::NetWeight => 0.22kg;
end WheelRotationSensor;
```

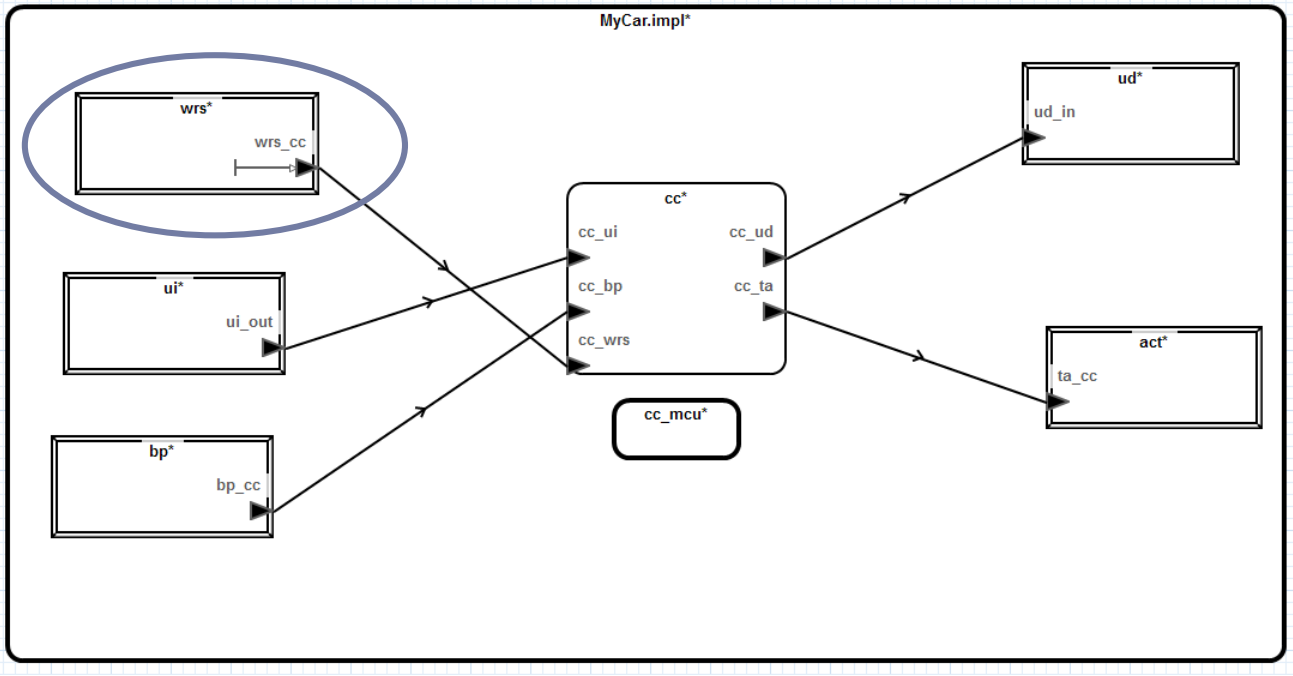
```
device ThrottleActuator
    features
        ta_cc: in data port;
    flows
        ta_sink: flow sink ta_cc {
            Latency => 15ms..15ms;
        };
    properties
        SEI::NetWeight => 0.3kg;
end ThrottleActuator;
```

```
system CruiseControl
    features
        cc_ui: in data port;
        cc_ud: out data port;
        cc_bp: in data port;
        cc_ta: out data port;
        cc_wrs: in data port;
    flows
        Flow_W_T: flow path cc_wrs -> cc_ta {
            Latency => 10ms..20ms;
        };
end CruiseControl;
```



Component Type: Flow Specification

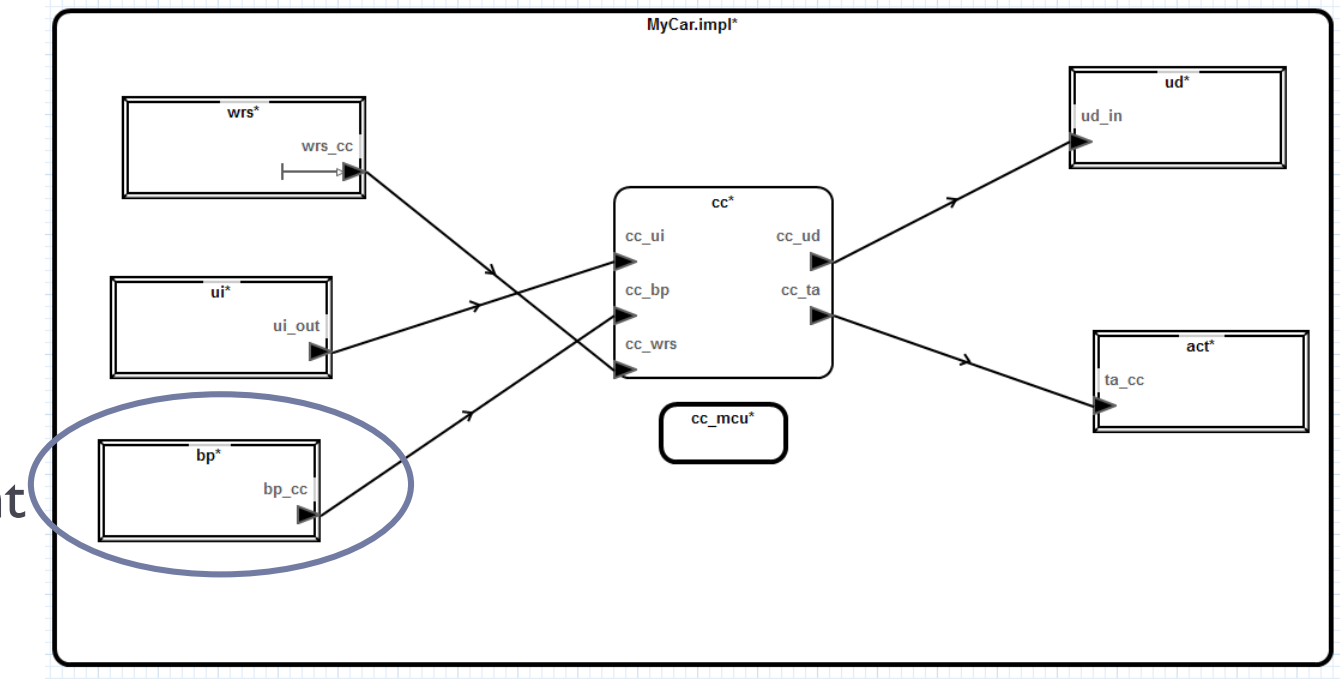
- Flow source
  - Specifies the origination of the flow
- Flow sink
  - Specifies the termination of the flow
- Flow path
  - Specifies the traversal of a component




```
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;
```

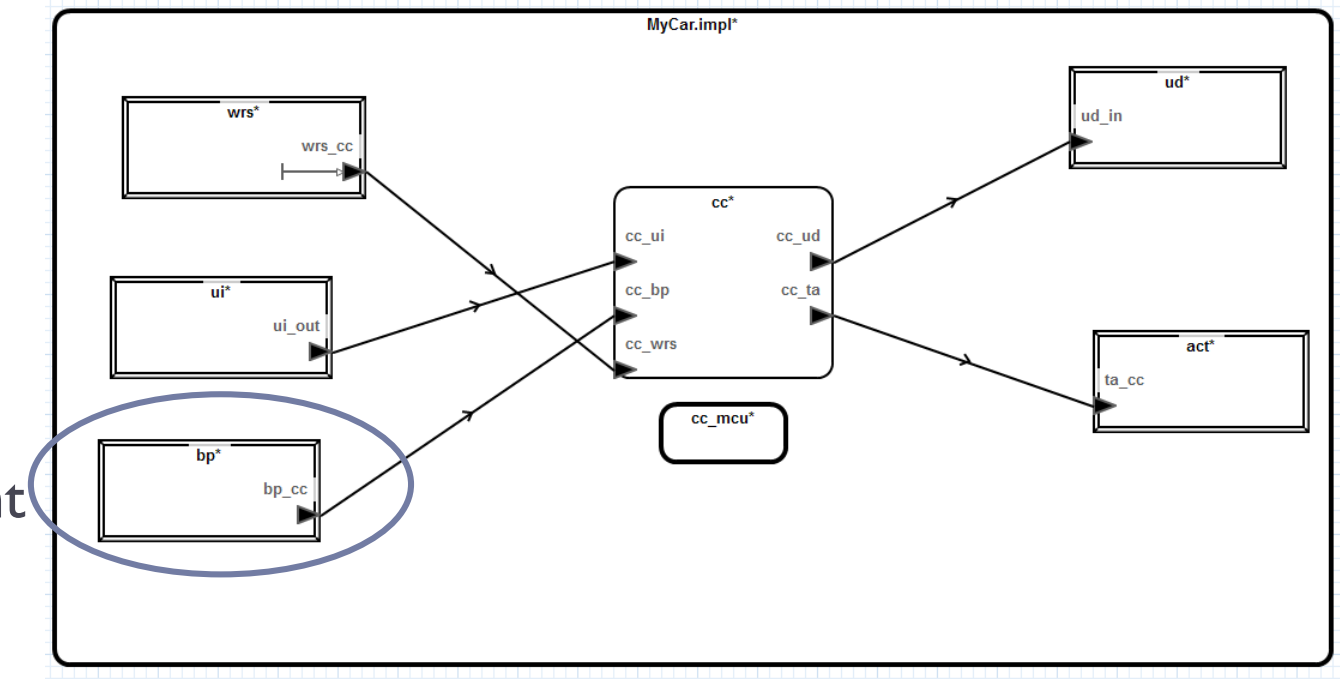
Component Type: Flow Specification (Exercise)

- Flow source
  - Specifies the origination of the flow
- Flow sink
  - Specifies the termination of the flow
- Flow path
  - Specifies the traversal of a component



Component Type: Flow Specification

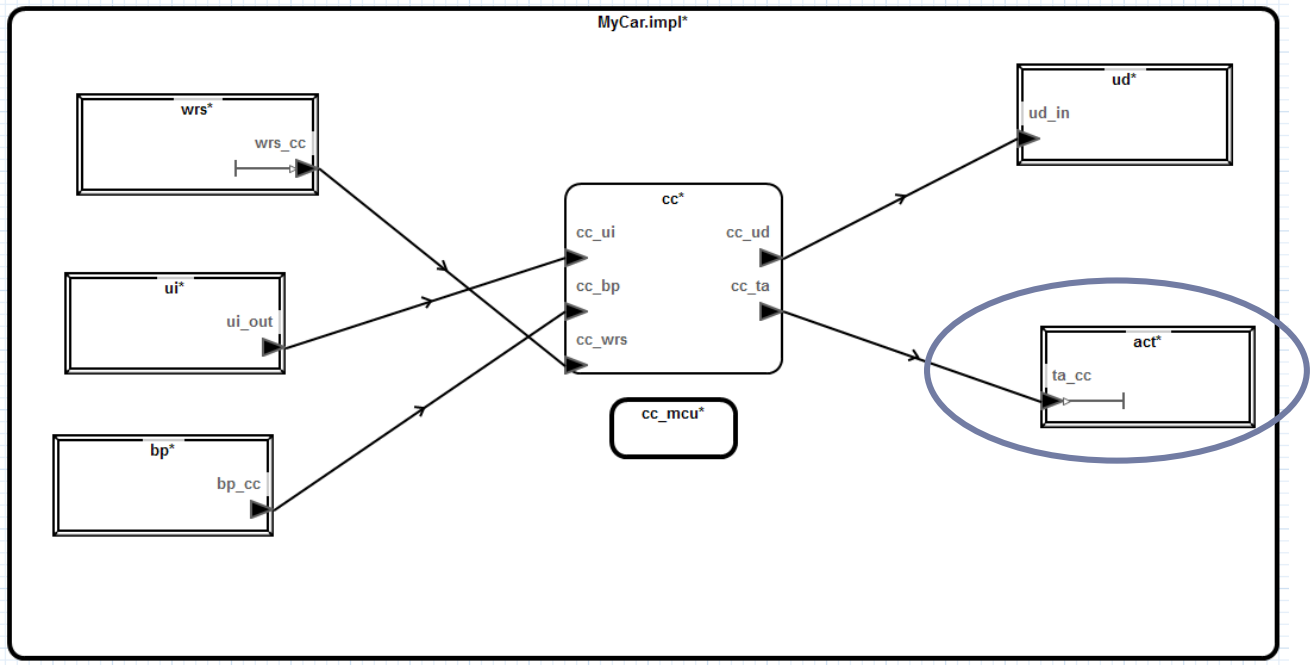
- Flow source
  - Specifies the origination of the flow
- Flow sink
  - Specifies the termination of the flow
- Flow path
  - Specifies the traversal of a component




```
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;
```

Component Type: Flow Specification

- Flow source
  - Specifies the origination of the flow
- Flow sink
  - Specifies the termination of the flow
- Flow path
  - Specifies the traversal of a component

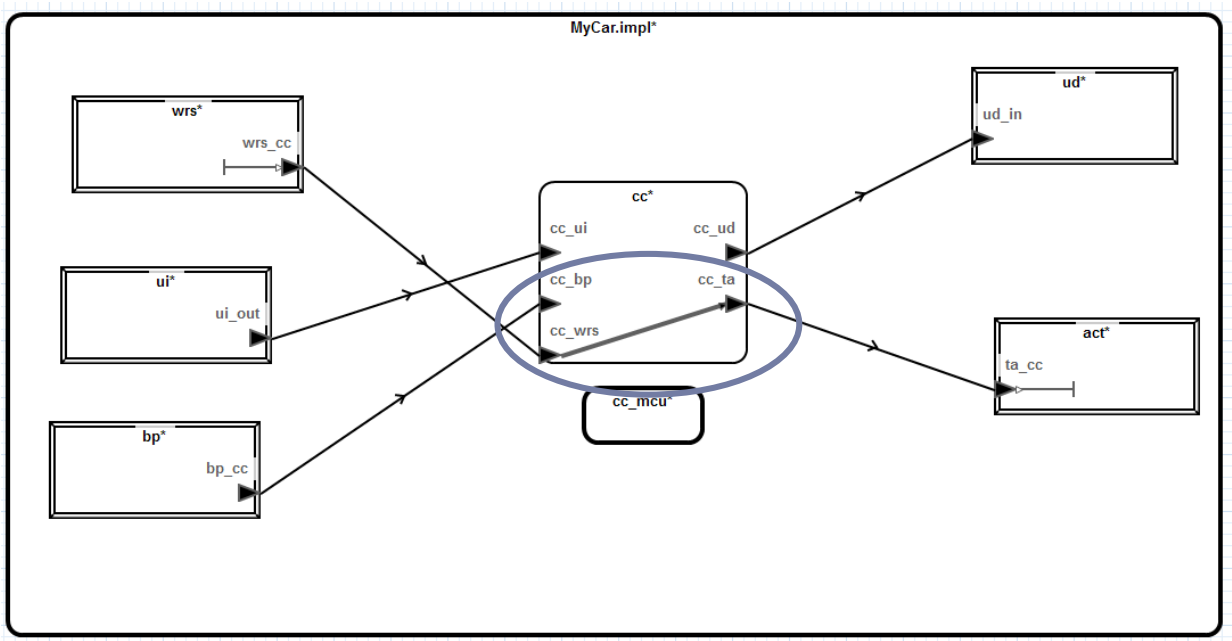



```
device ThrottleActuator
    features
        ta_cc: in data port;
    flows
        ta_sink: flow sink ta_cc {
                    Latency => 15Ms .. 15Ms;
                };
    properties
        SEI::NetWeight => 2.0kg;
end ThrottleActuator;
```

Component Type: Flow Specification

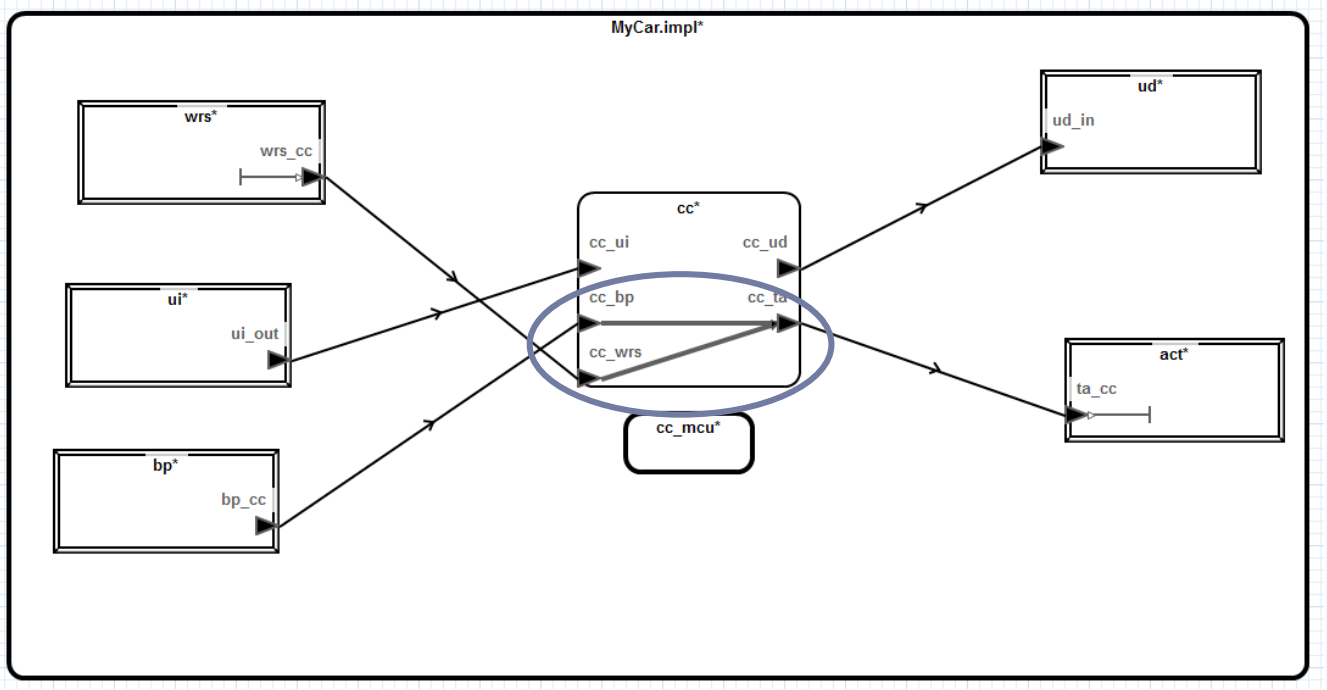
- Flow source
  - Specifies the origination of the flow
- Flow sink
  - Specifies the termination of the flow
- Flow path
  - Specifies the traversal of a component

```
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 .. 20Ms;
                };

end CruiseControl;
```



Component Type: Flow Specification (Exercise)

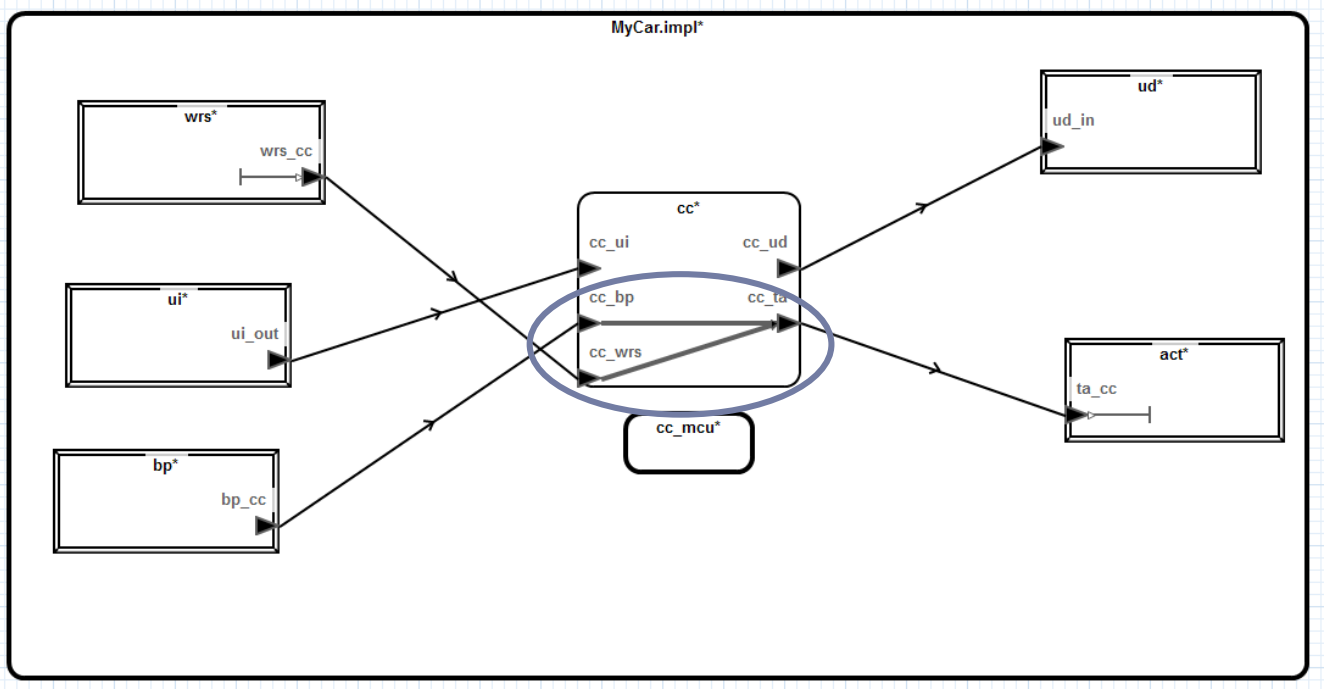
- Flow source
  - Specifies the origination of the flow
- Flow sink
  - Specifies the termination of the flow
- Flow path
  - Specifies the traversal of a component



Component Type: Flow Specification

- Flow source
  - Specifies the origination of the flow
- Flow sink
  - Specifies the termination of the flow
- Flow path
  - Specifies the traversal of a component

```
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 .. 20Ms;
                };
        Flow_B_T: flow path cc_bp -> cc_ta {
                    Latency => 15Ms .. 15Ms;
                };
end CruiseControl;
```



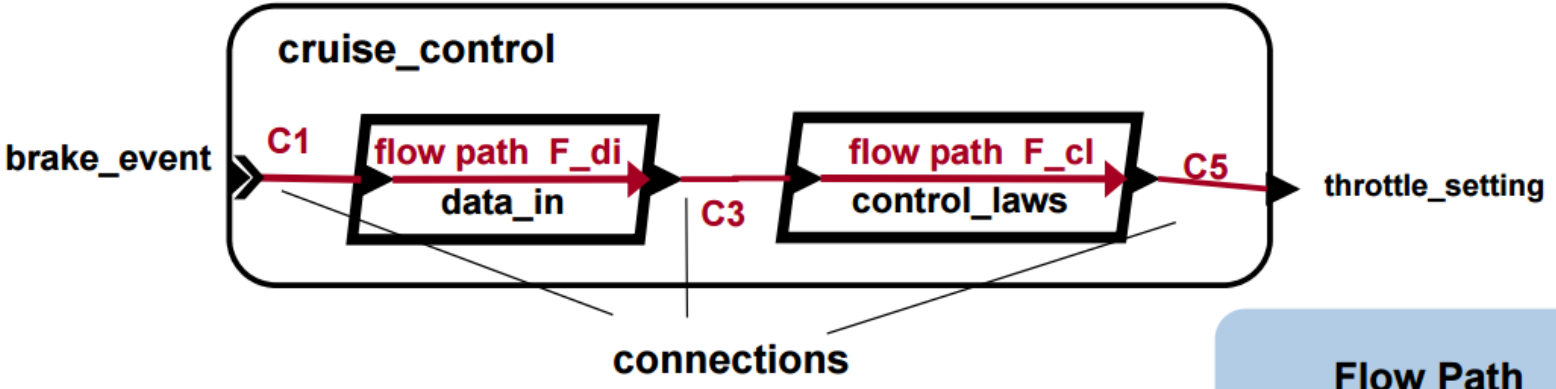
Component Implementation: Flow Implementation

Flow through subcomponents and connections

Subcomponent flow in terms of its flow specification

Flow Path Specification

```
brake_flow: flow path brake_event -> throttle_setting;
```



Flow Path Implementation

```
brake_flow: flow path brake_event ->
            C1 -> data_in.F_di ->
            C3 -> control_laws.F_cl ->
            C5 -> throttle_setting;
```

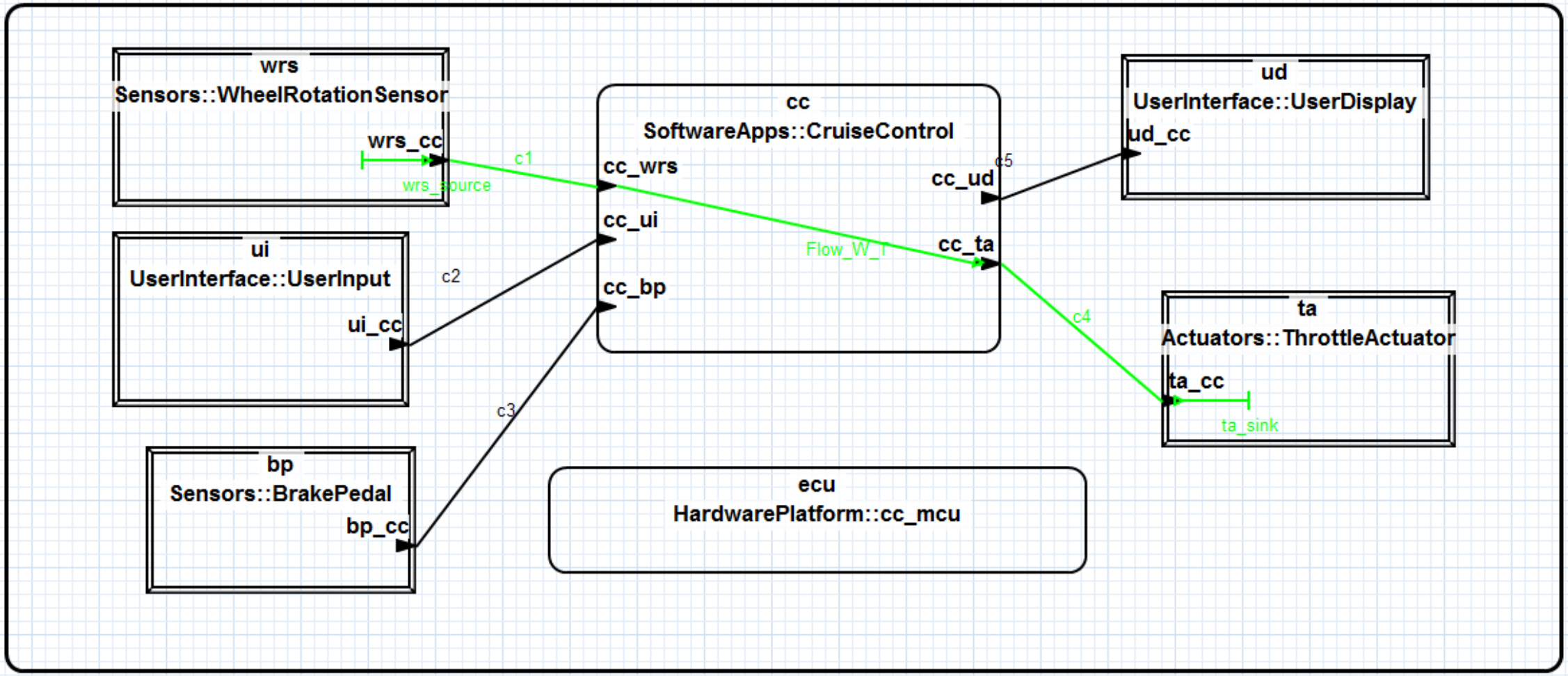
component_name.flow_name

Component Type: Flow Specification and Implementation (Exercise)

推荐
modeler.org.cn

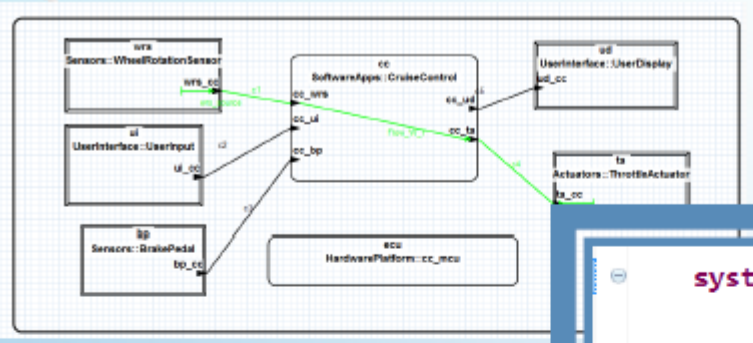
Component Implementation: End-to-End Flow

推荐
modeler.org.cn



Component Implementation: End-to-End Flow

推荐
modeler.org.cn




```
system implementation MyCar.impl
    subcomponents
        ui: device UserInterface::UserInput;
        ud: device UserInterface::UserDisplay;
        cc: system SoftwareApps::CruiseControl;
        ecu: system HardwarePlatform::cc_mcu;
        ta: device Actuators::ThrottleActuator;
        wrs: device Sensors::WheelRotationSensor;
        bp: device Sensors::BrakePedal;
    connections -- Port to port connections
        c1: port wrs.wrs_cc -> cc.cc_wrs;
        c2: port ui.ui_cc -> cc.cc_ui;
        c3: port bp.bp_cc -> cc.cc_bp;
        c5: port cc.cc_ud -> ud.ud_cc;
        c4: port cc.cc_ta -> ta.ta_cc;
flows
        Flow_W_T: end to end flow wrs.wrs_source -> c1 -> cc.Flow_W_T -> c4 -> ta.ta_sink {
            Latency => 40ms..40ms;
        };
end MyCar.impl;
```

End-To-End Flow Latency Analysis

推荐
modeler.org.cn