

Software Architecture in Action

Flavio Oquendo, Jair C Leite, Thais Batista



Chapter 7

Executing Software Architectures

Learning outcomes of this chapter

- You will learn:
 - the *execution semantics* of an architecture description expressed with SysADL;
 - the *execution semantics* of components, ports, connectors, and configurations;
 - the *execution semantic* of activities, actions and related elements.

The structure of this chapter

- Introduction
- Executing an architecture
- Executing an activity
- Example
- Summary
- For further reading



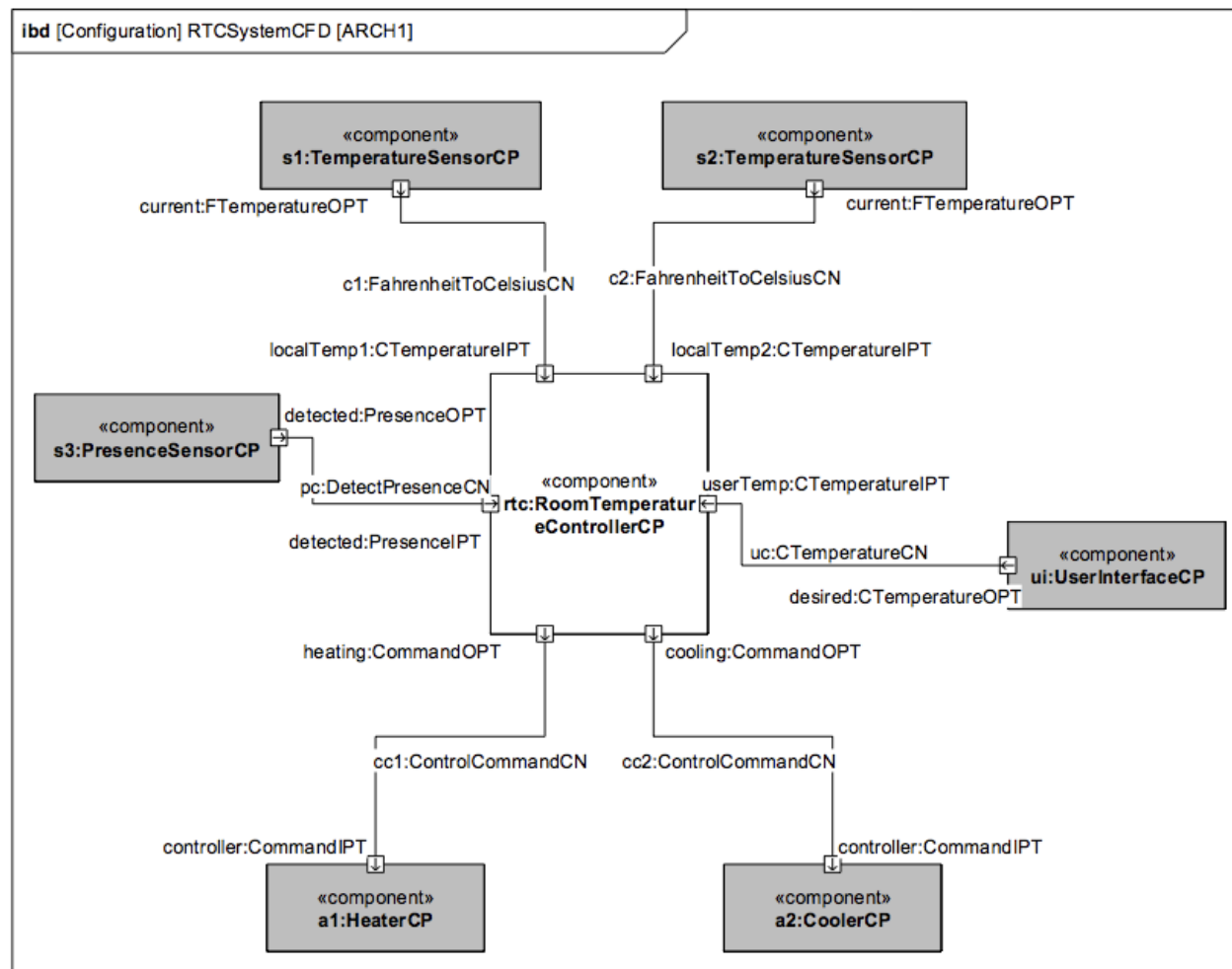
Introduction

Introduction (1/2)

- The *executable viewpoint* adds the executable elements of the architecture description enabling to execute architecture descriptions
 - We address the operational semantics of SysADL showing the execution of its elements
 - The operational semantics is illustrated through our running example of the *RTC* system

Introduction (2/2)

■ Configuration of the *RTC* system





Executing an architecture

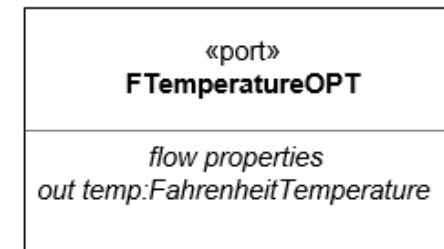
Executing an architecture

Executing boundary components

Execution semantics

- Considering the execution perspective, a component is a running process, seen as a black-box communicating with its environment through ports
- Execution semantics of boundary components
 - The execution of a boundary component is given by the communication via its ports in conformance to the allocated protocol

SysADL

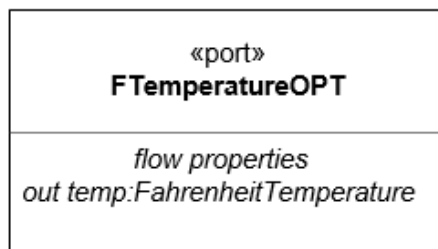


Executing an architecture

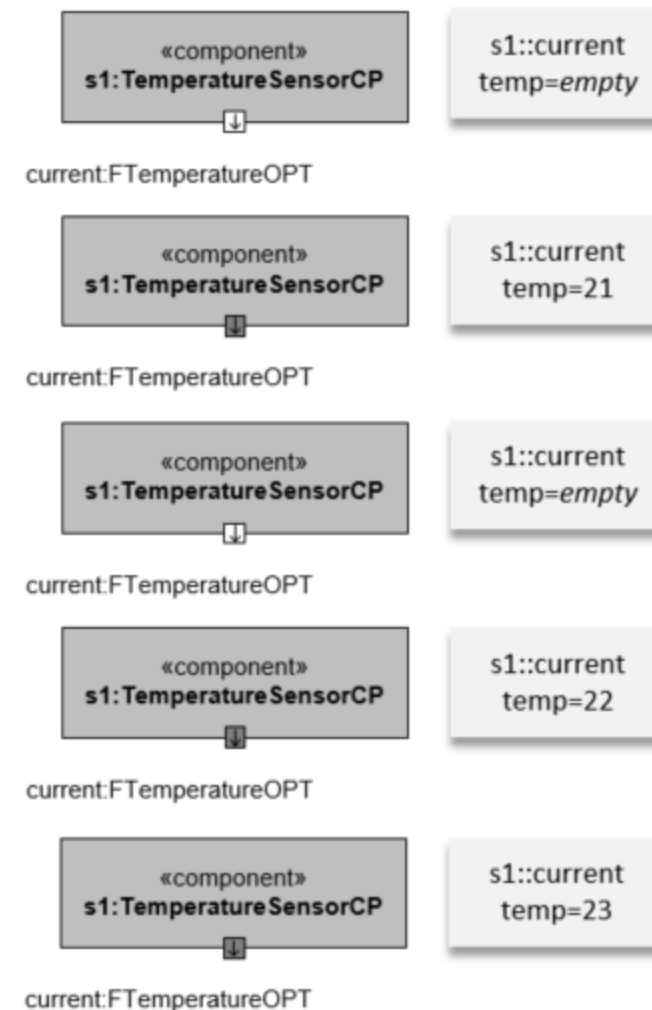
Executing boundary components

Execution semantics

- Execution semantics of boundary components
 - illustrating how the execution of the *s1* component is given by the protocol of the *FTemperatureOPT* port
 - that protocol specifies that the port sends Fahrenheit temperatures in a flow



SysADL



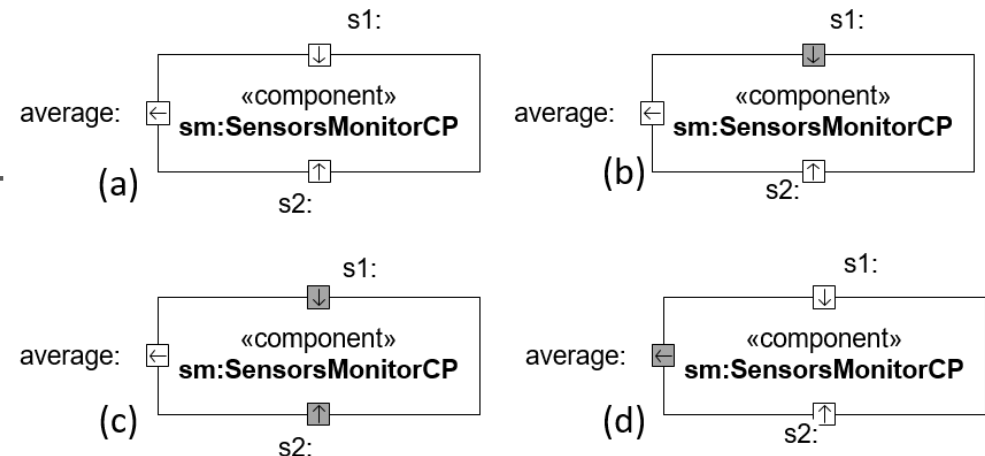
Executing an architecture

Executing simple components

Execution semantics

- Considering the execution perspective, a component is a running process, seen as a black-box communicating with its environment through ports
- Execution semantics of simple components
 - The execution semantics of a simple component is given by the execution of its behavior as specified in an associated activity

SysADL



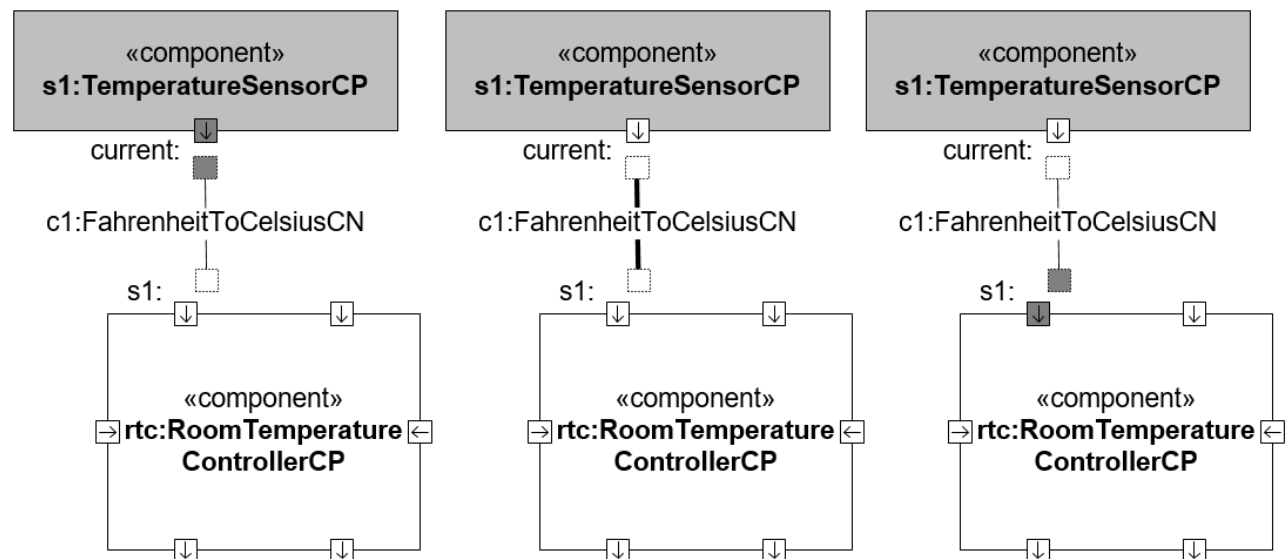
Executing an architecture

Executing connectors

Execution semantics

SysADL

- Execution semantics of connectors
 - The execution semantics of a connector is given by the flow direction specification between its participant ports



Executing an architecture

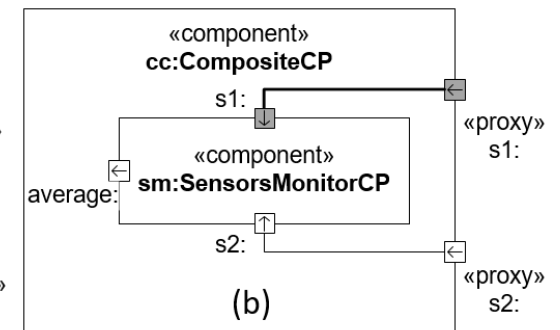
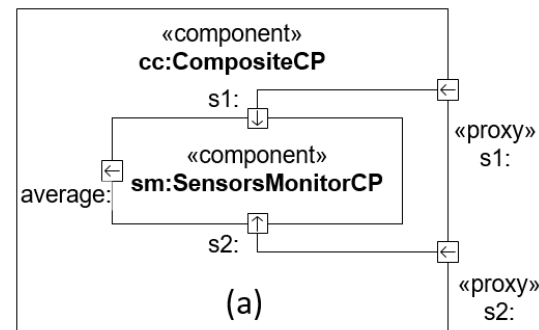
Executing delegations

Execution semantics

■ Execution semantic of delegations

- Each port of a composite component has a delegation to some of the ports of its internal component
- The external port acts as a proxy
- Whenever a value arrives in the proxy port of the composite component, it is immediately sent to the corresponding port

SysADL



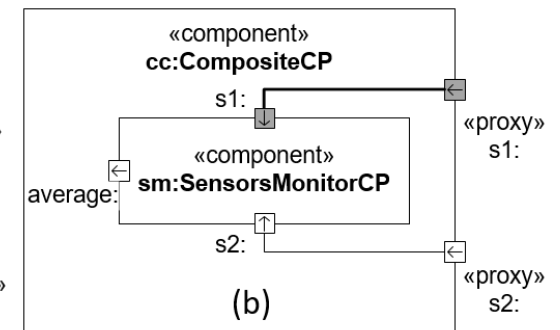
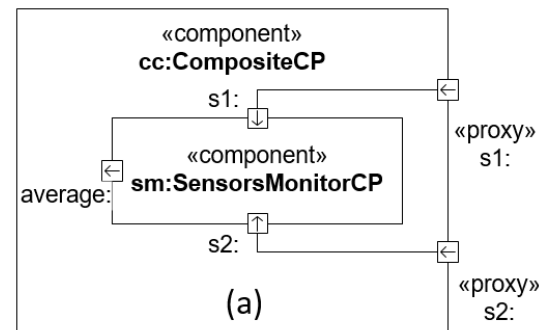
Executing an architecture

Executing configurations

Execution semantics

- Execution semantic of composite components
 - The execution of a composite component is given by the concurrent composition of the behavior of the interconnected sub-components
 - When a component has input ports connected to an external proxy ports its execution may begin immediately when the proxy ports receive their values
 - Then the execution follows according to the interconnected sub-components

SysADL





Executing an activity

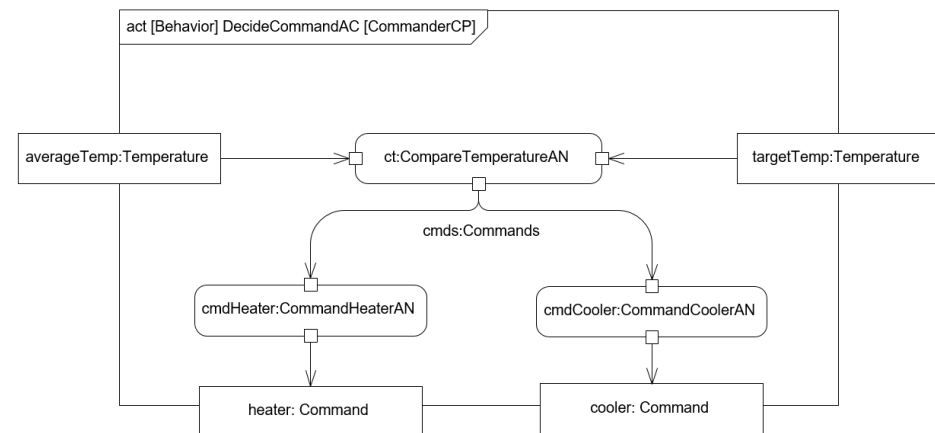
Executing an activity

Executing an activity

Execution semantics

- **Activities** are the core elements to specify the behavior of component, connectors and configuration, their operational semantics defines the core rules to the execution of an architecture description
- The elements in an activity are flows, delegations, decisions, buffers, data stores, and actions
- Semantics rules are expressed by premises and conclusions

SysADL




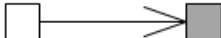
Executing an activity

Executing an activity

Execution semantics

SysADL

- **Flow:** a flow is a specification that a value can flow from an element (except parameters, delegations or actions) to another element (except parameters, delegations or actions)
- The semantic of a flow is that when there is a value in an element it flows to the element in the other element

Name	Premise	Conclusion
<i>Flow</i>		



Executing an activity

Executing an activity

Execution semantics

SysADL

- **Delegation:** a delegation specifies that a value in a parameter is directly assigned in the pin linked to it
- The semantic of a delegation is that when there is a value in an element it assigns to the element in the other element

Name	Premise	Conclusion
<i>Delegation</i>		

Executing an activity

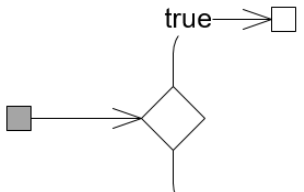
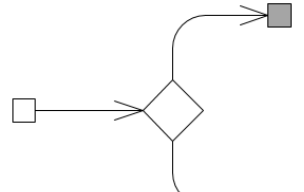
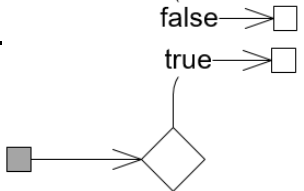
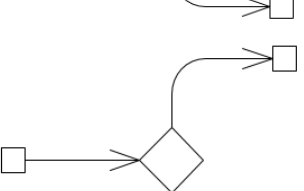
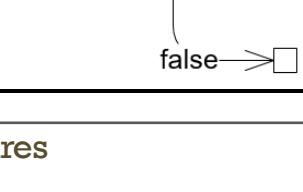
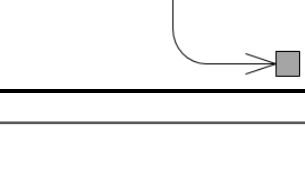
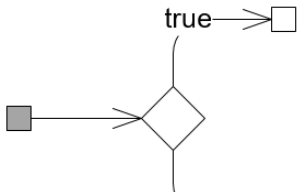
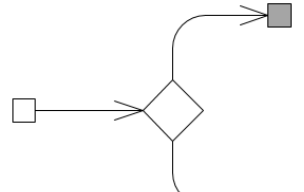
Executing an activity

Execution semantics

SysADL

■ Decision

- The semantic of a decision states that when there is a value in a pin that is part of a decision element, if the condition is true, it flows to the true side of the decision; if not it flows to the false side

Name	Premise	Conclusion
<i>Decision</i>		
		
		
		

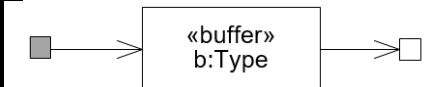
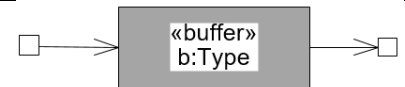
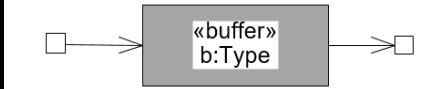
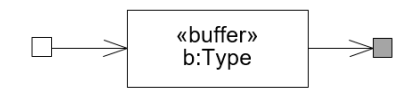
Executing an activity

Executing an activity

Execution semantics

SysADL

- **Buffer:** a buffer is an element that stores a value until it is consumed by another element
 - The semantic of a buffer states that when there is a value in a pin, it flows to the buffer
 - The buffer stores it until it is consumed by another element that has a pin connected to it

Name	Premise	Conclusion
<i>Buffer</i>		
		


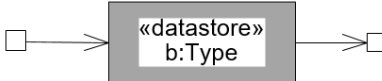
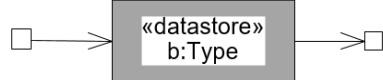
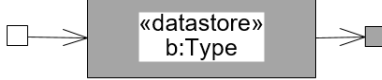
Executing an activity

Executing an activity

Execution semantics

SysADL

- **Datastore:** a datastore is an element that stores a value even when it is consumed by another element
- the semantic of a datastore is states that when there is a value in a pin, it flows to the datastore
- It stores the value even if it is consumed by another element that has a pin connected to it

Name	Premise	Conclusion
<i>Datastore</i>		
		

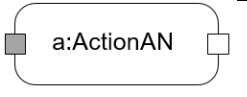
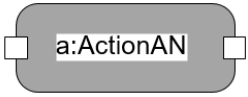
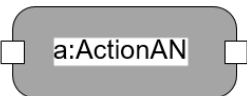
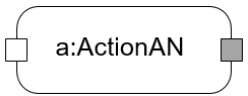
Executing an activity

Executing an activity

Execution semantics

SysADL

- **Action:** An action is a behavioral element that executes a body of statements using the input pins (parameters of the action) and returns a value to an output pin
- When there is a value in the input pin (premise), the action begins to execute (conclusion)
- In the execution of an action (premise), the result is provided in the out pin (conclusion)

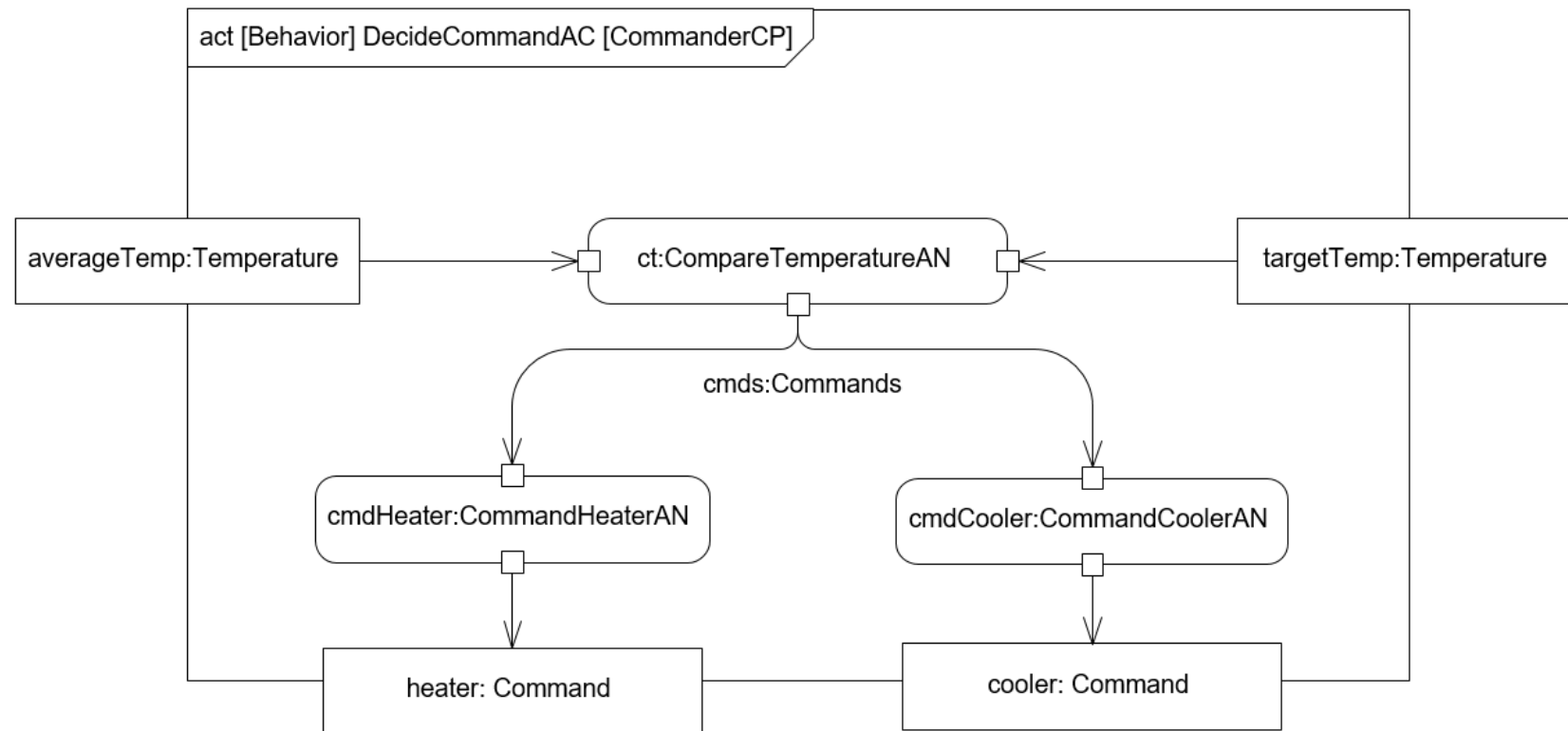
Name	Premise	Conclusion
<i>Action</i>		
		

Executing an activity

Executing an activity

Execution semantics

■ Execution step 1

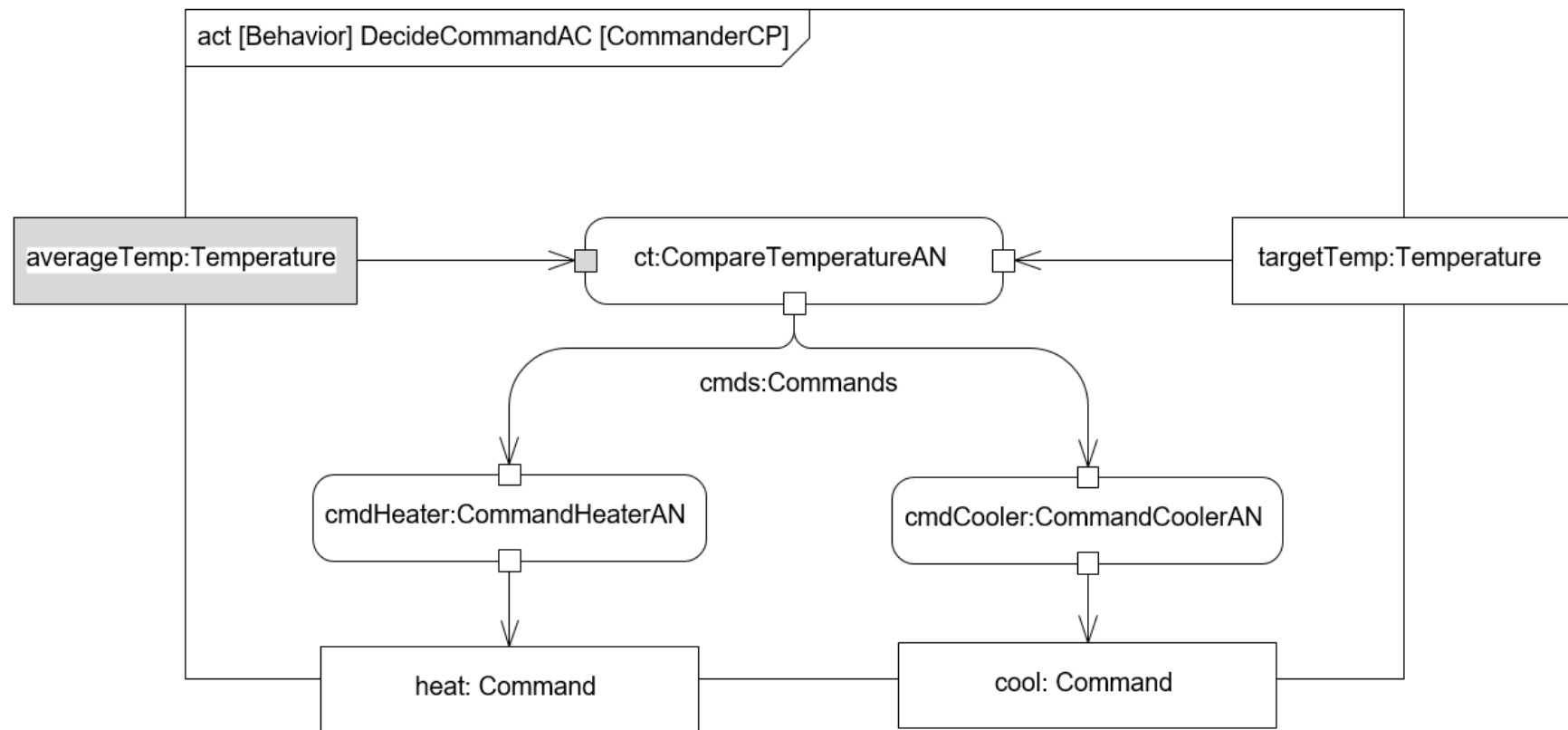


Executing an activity

Executing an activity

Execution semantics

■ Execution step 2

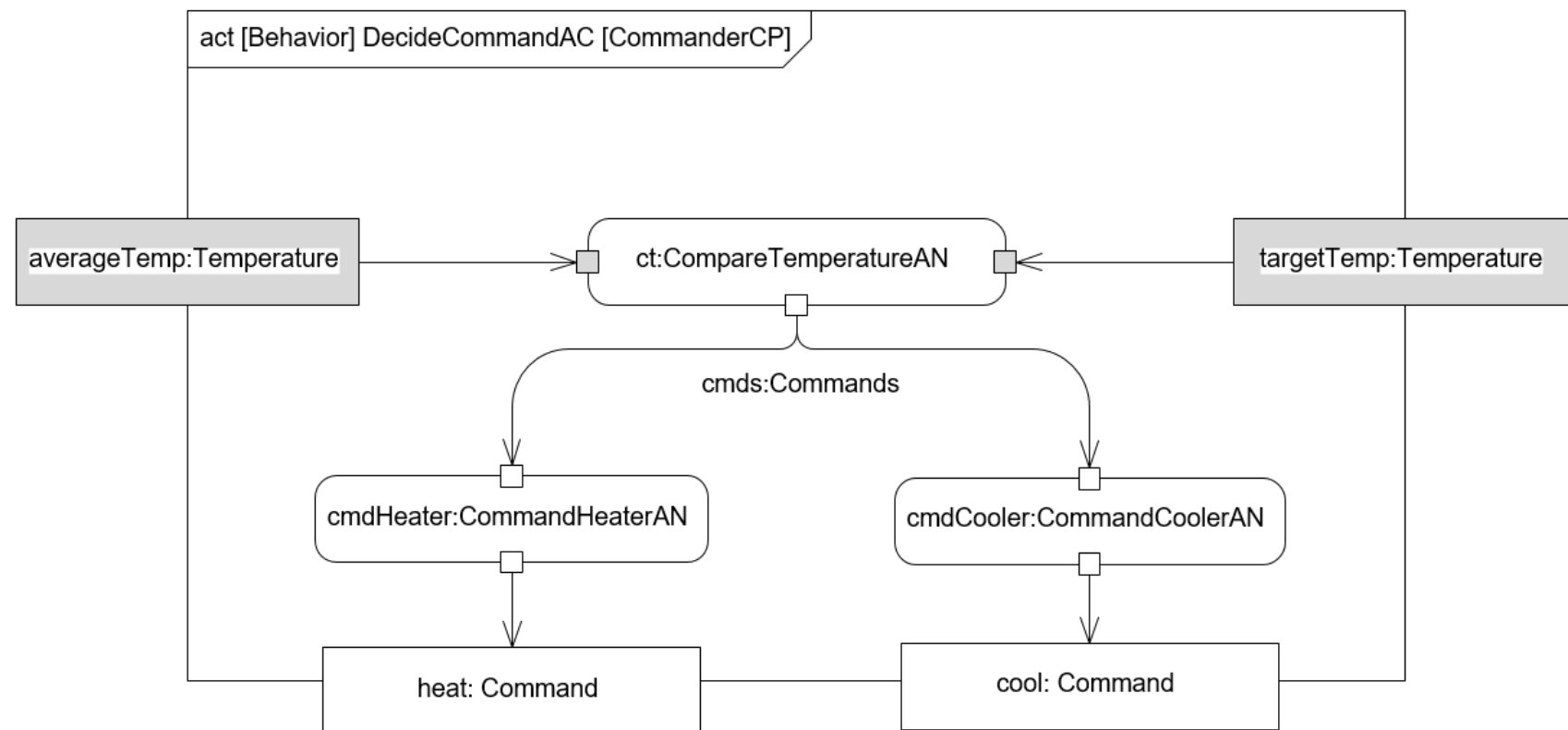


Executing an activity

Executing an activity

Execution semantics

■ Execution step 3

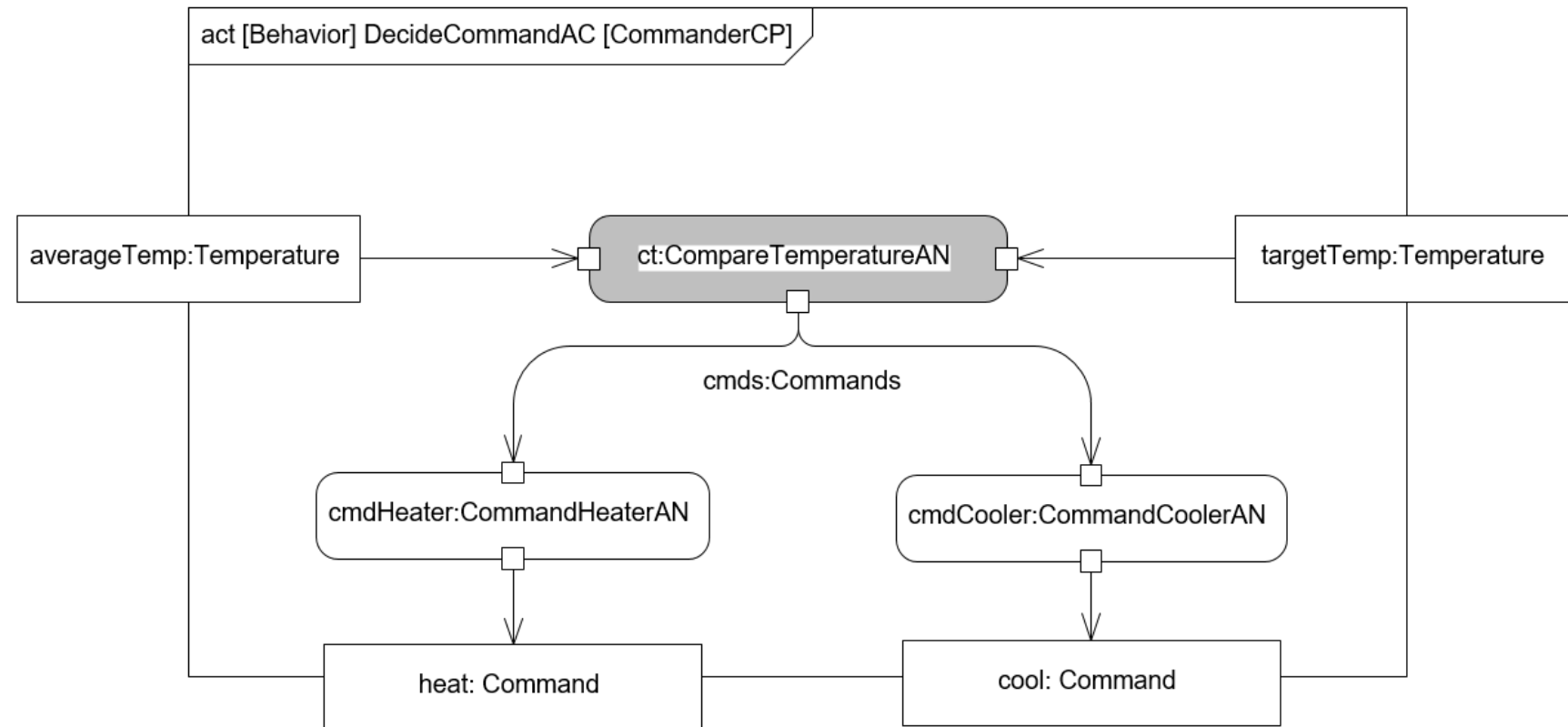


Executing an activity

Executing an activity

Execution semantics

■ Execution step 4

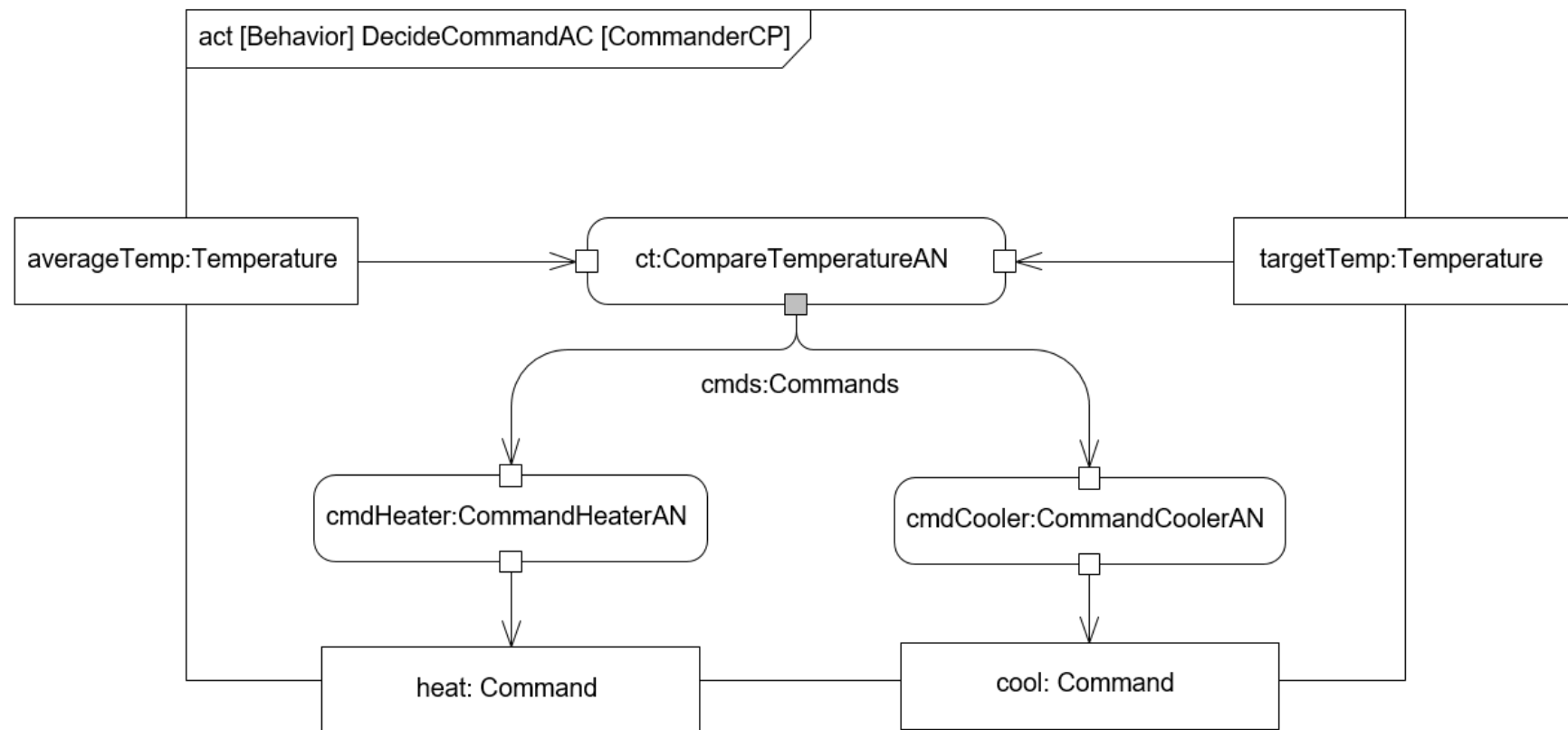


Executing an activity

Executing an activity

Execution semantics

■ Execution step 5

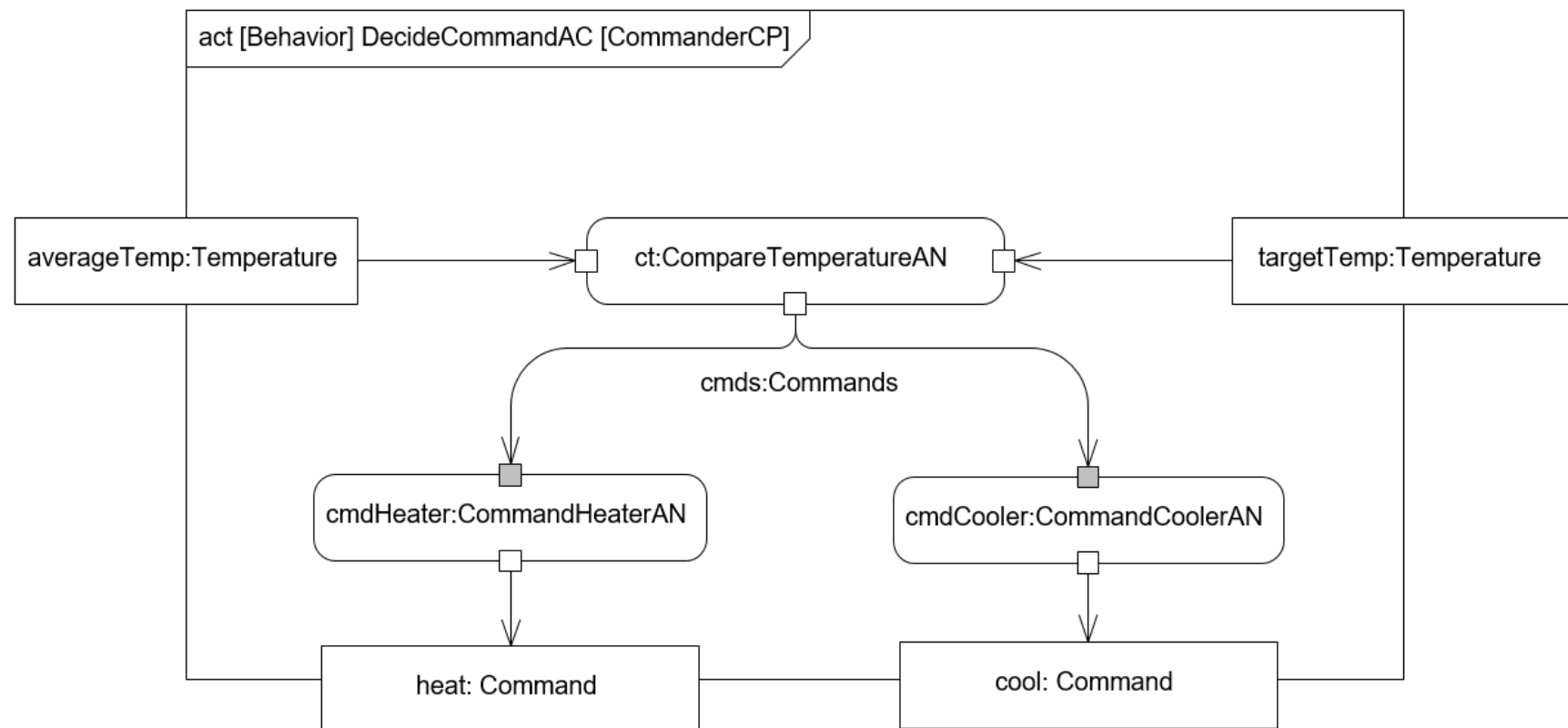


Executing an activity

Executing an activity

Execution semantics

■ Execution step 6

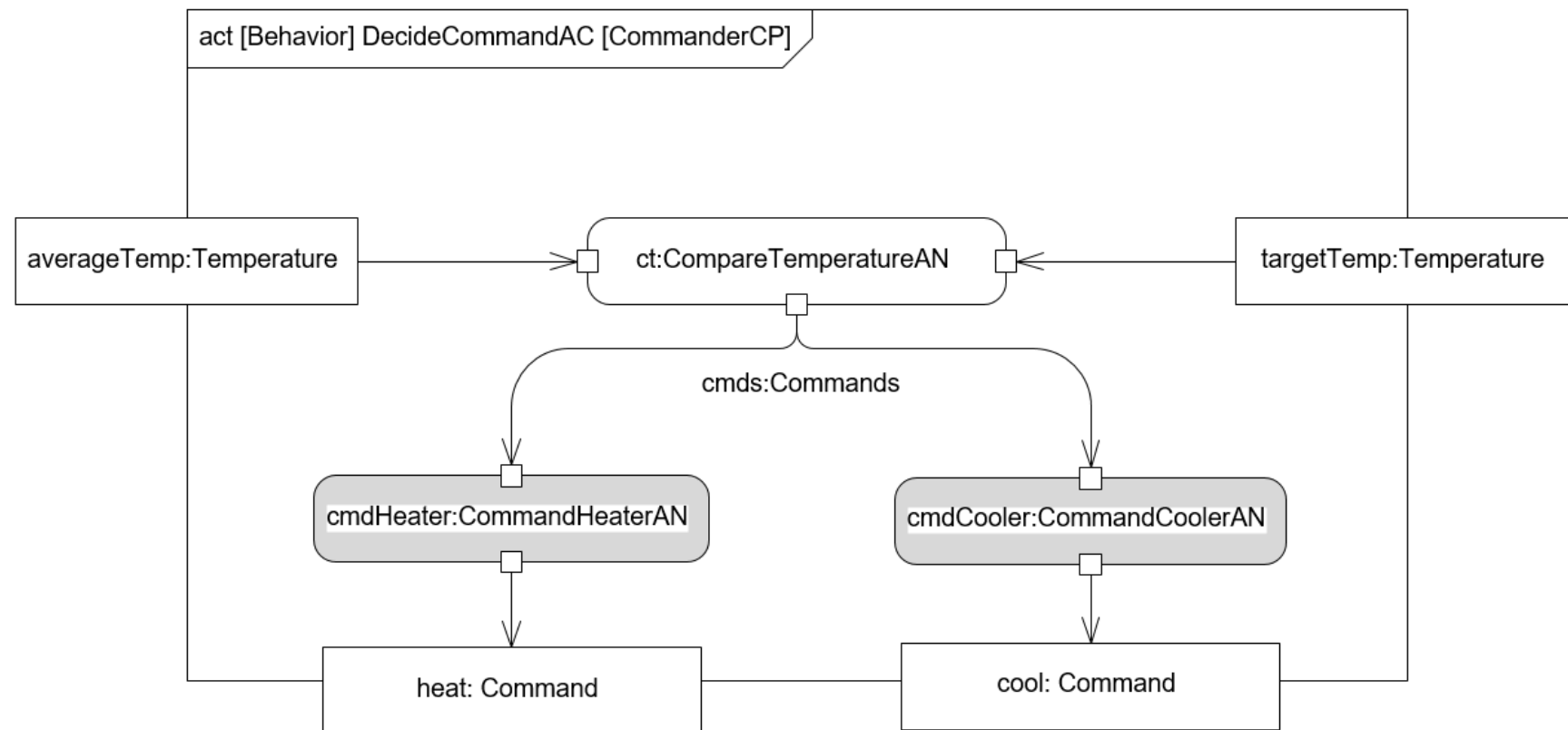


Executing an activity

Executing an activity

Execution semantics

■ Execution step 7

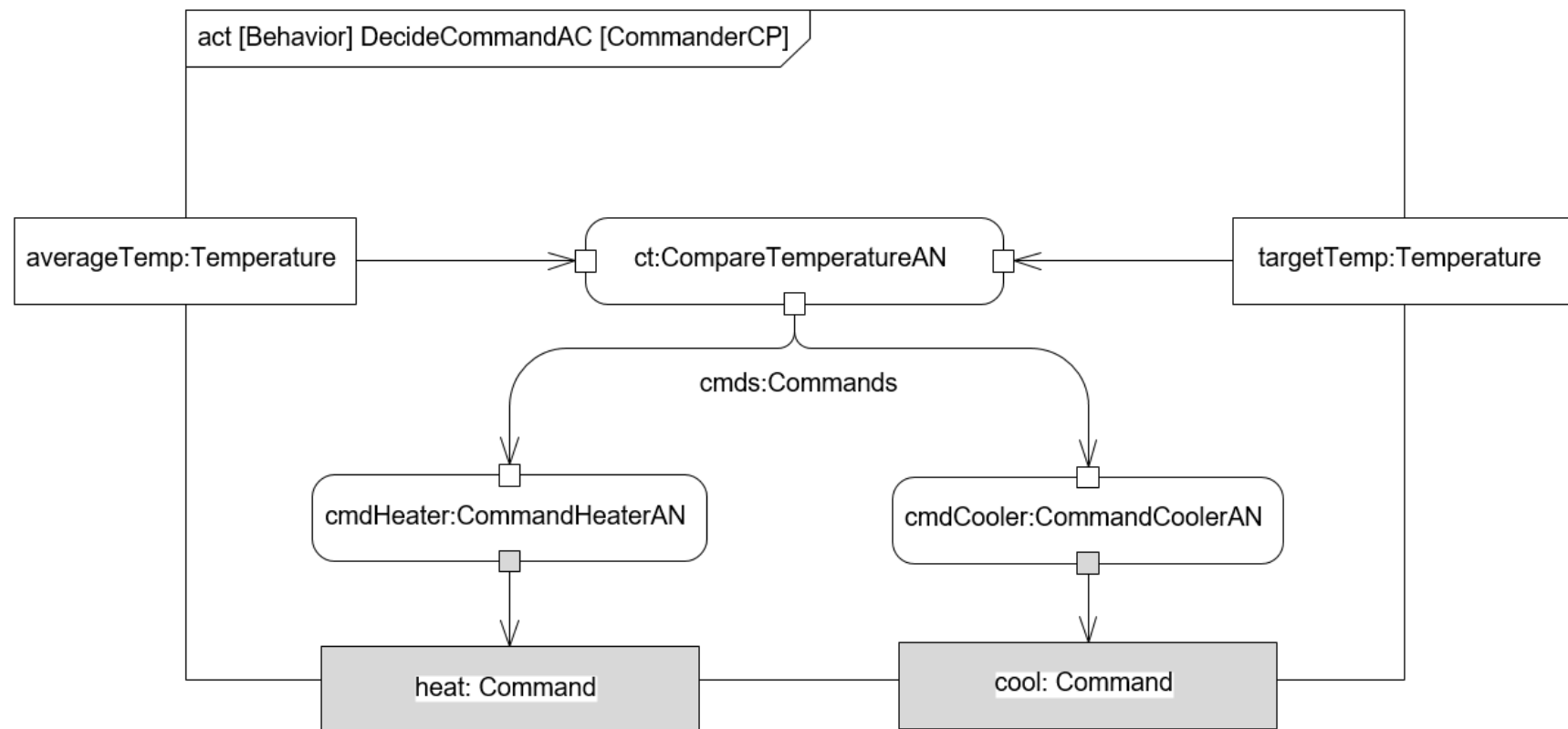


Executing an activity

Executing an activity

Execution semantics

■ Execution step 8





Example

Example of architectural execution

Executing an architecture

Execution semantics

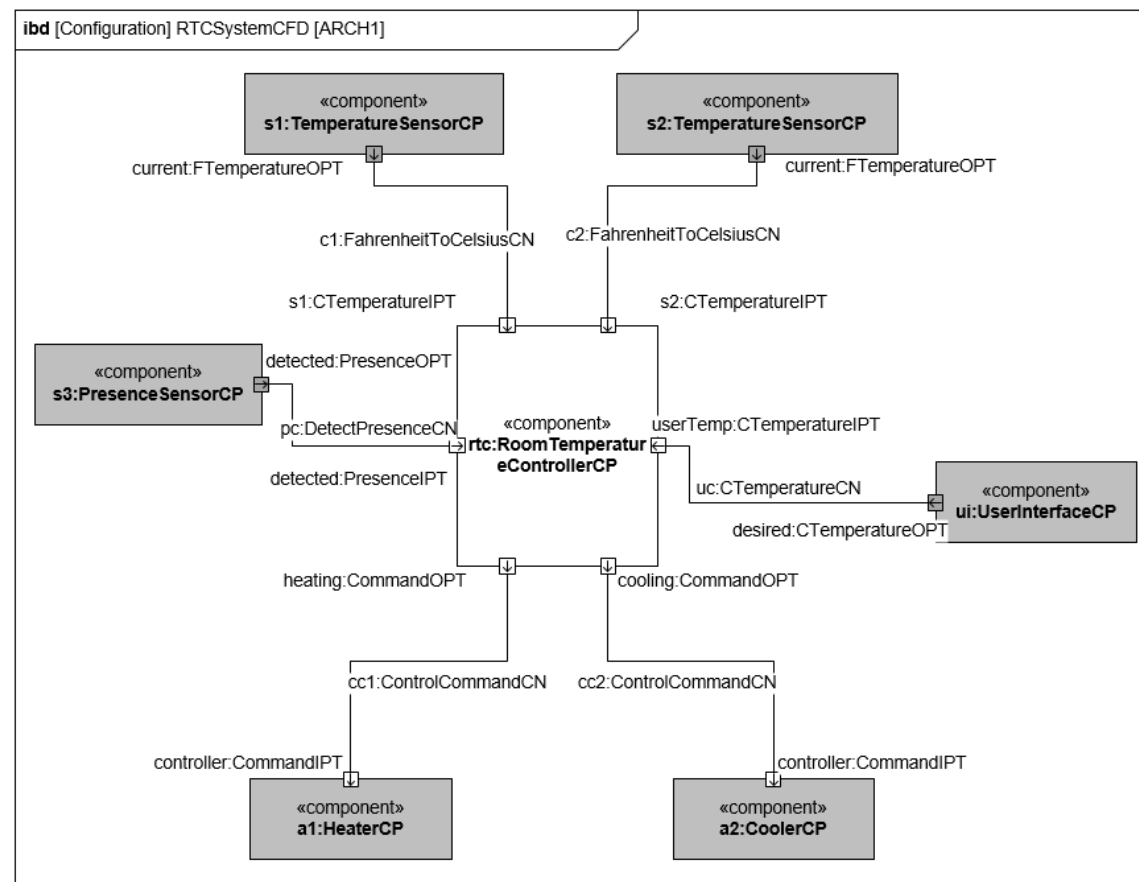
- Scenario of architecture execution to illustrate a running architecture
 - In this scenario, the system is installed in a single room with two temperature sensors in different locations
 - We consider the initial state of the running system a situation in which both the cooler and the heater are turned off, the room temperature is 80 degrees Fahrenheit in average, and a person is in the room and set the temperature to 25 degrees Celsius using the remote control
 - The temperature sensors continuously provide the values of the current temperature

Example of architectural execution

Executing an architecture

Execution semantics

■ Execution step 1 (1st level)

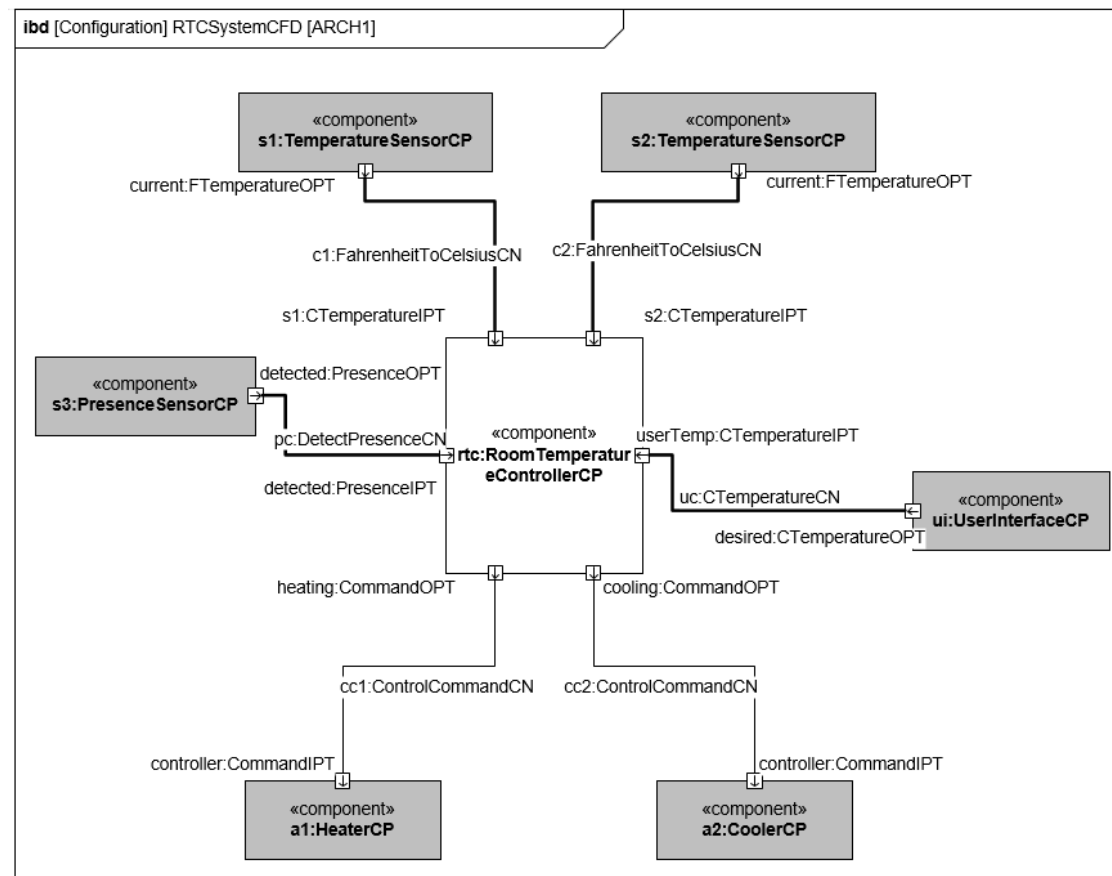


Example of architectural execution

Executing an architecture

Execution semantics

■ Execution step 2 (1st level)

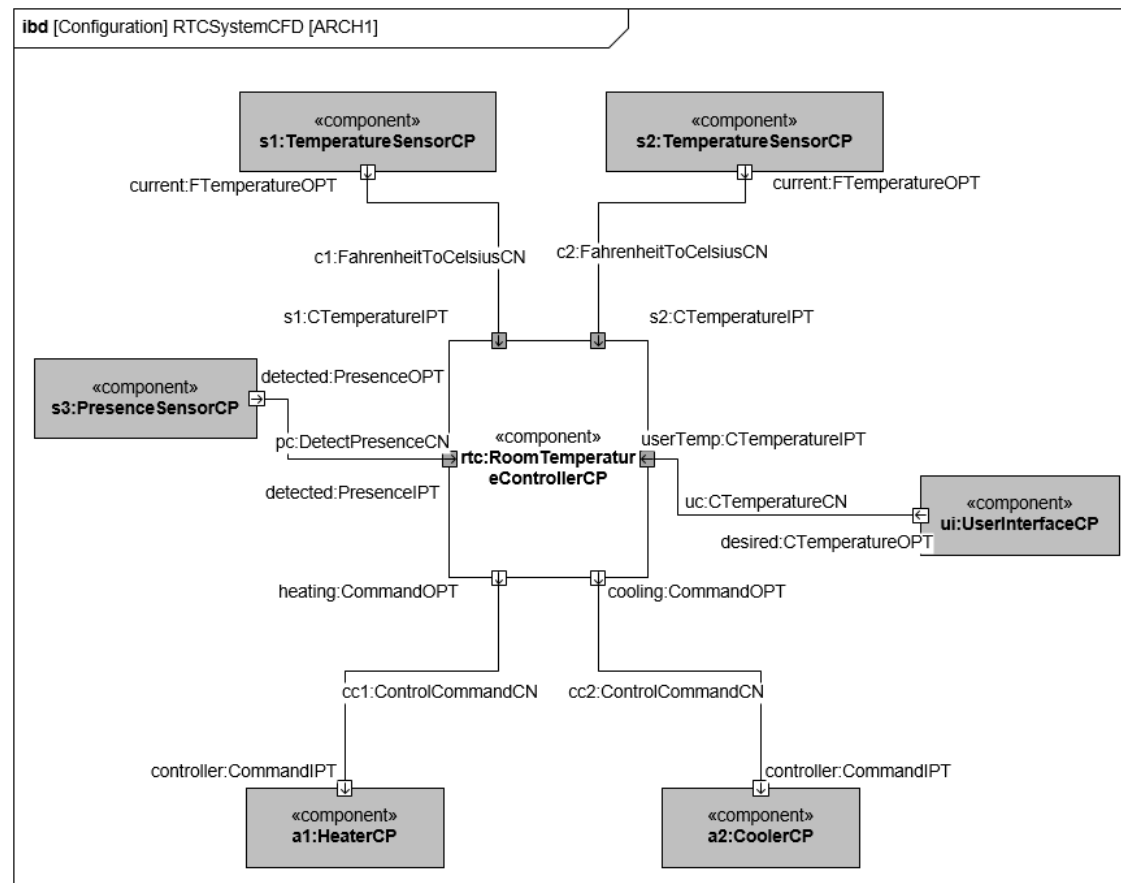


Example of architectural execution

Executing an architecture

Execution semantics

■ Execution step 3 (1st level)

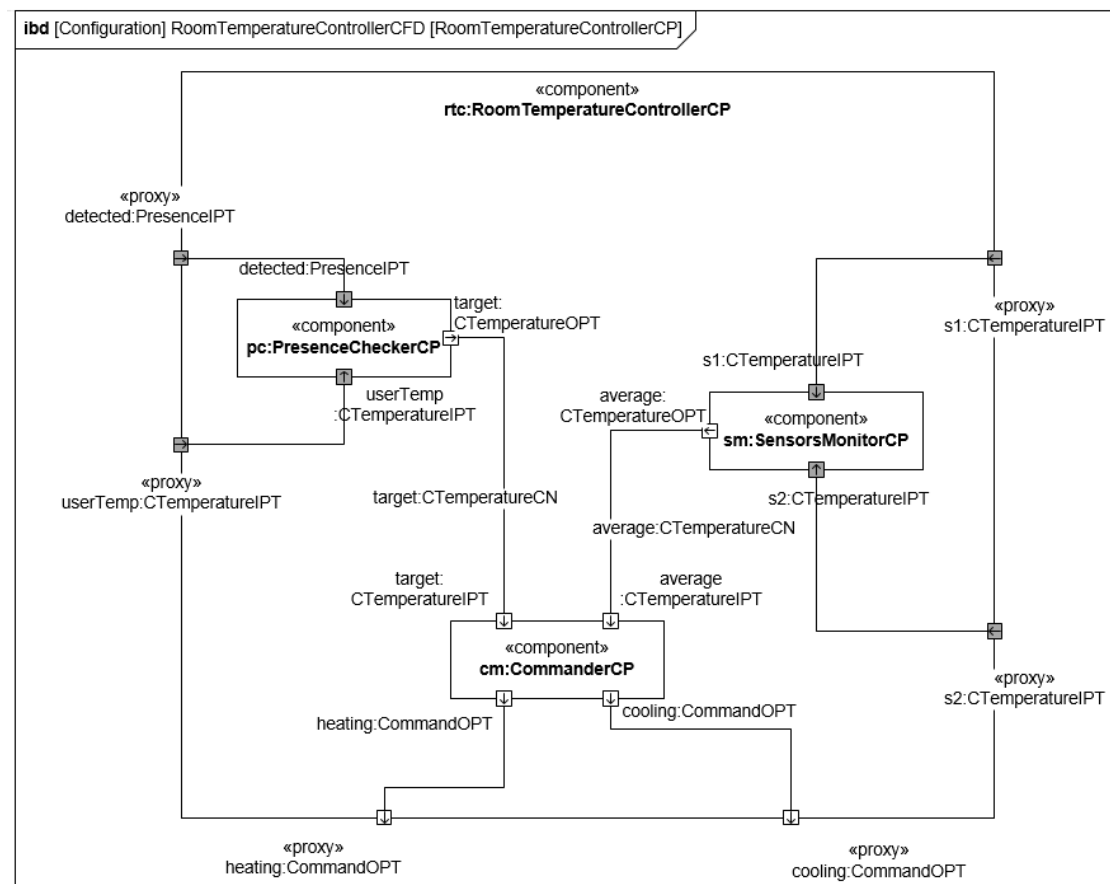


Example of architectural execution

Executing an architecture

Execution semantics

■ Execution step 4 (1st inside 2nd level)

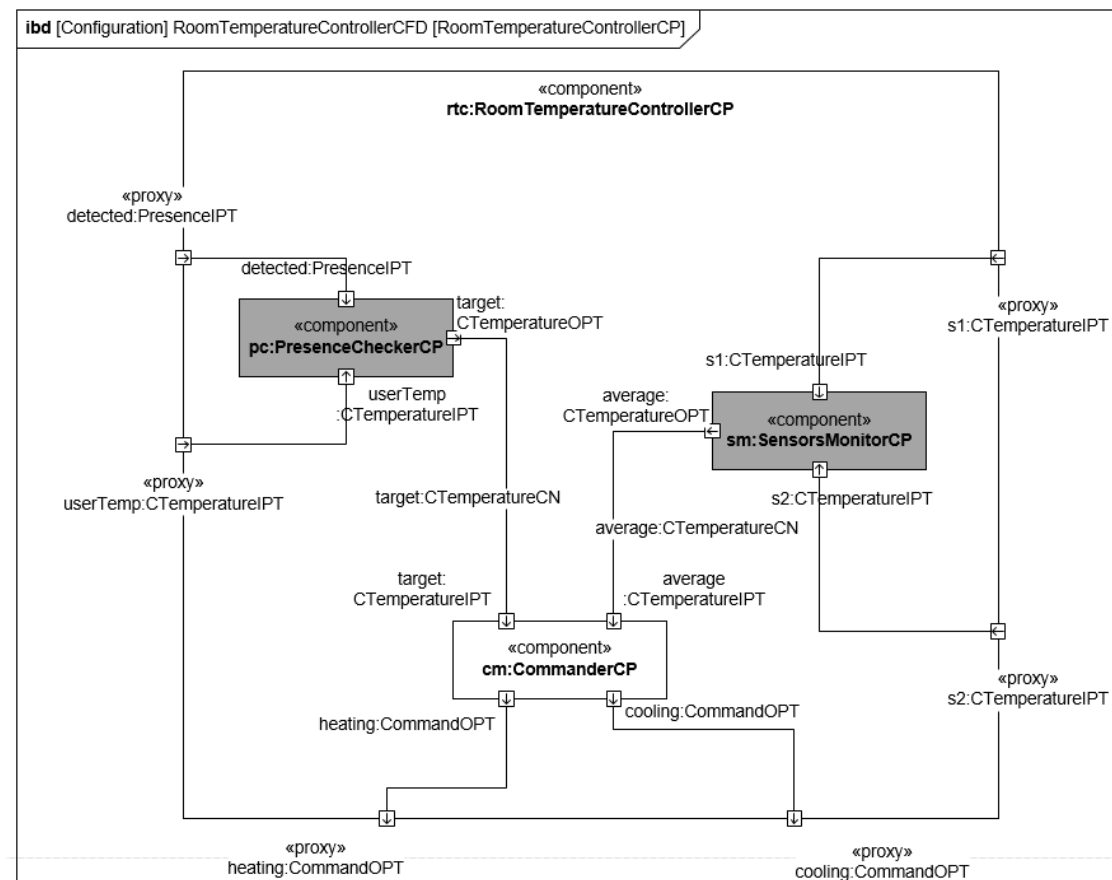


Example of architectural execution

Executing an architecture

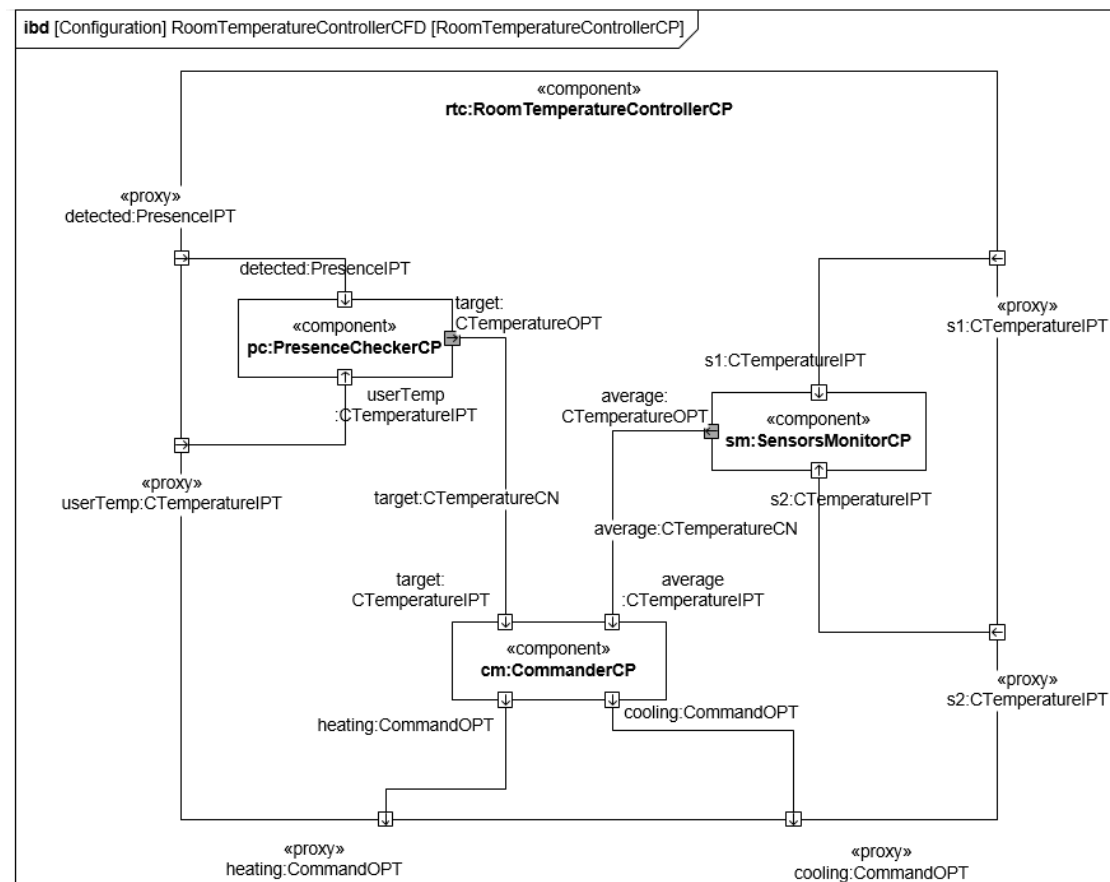
Execution semantics

■ Execution step 5 (2nd inside 2nd level)



Execution semantics

- Execution step 6 (3rd inside 2nd level)

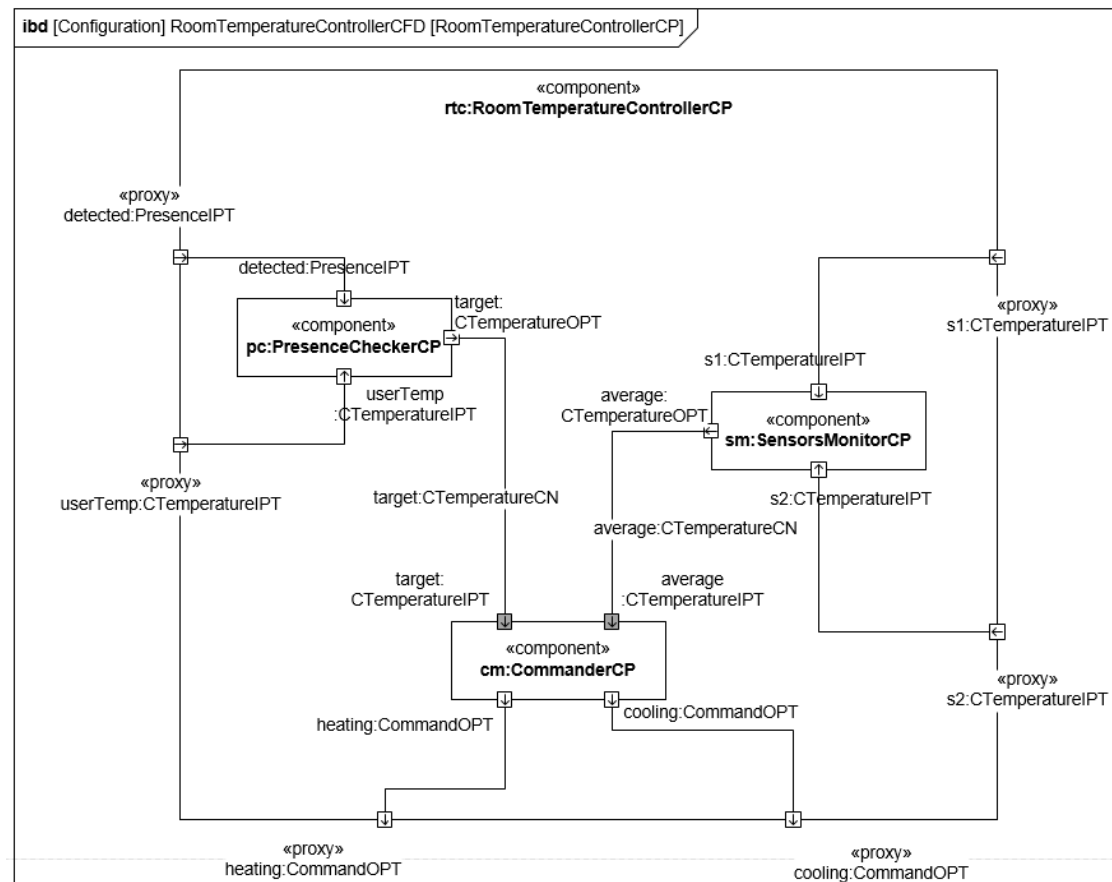


Example of architectural execution

Executing an architecture

Execution semantics

■ Execution step 7 (4th in inside 2nd level)

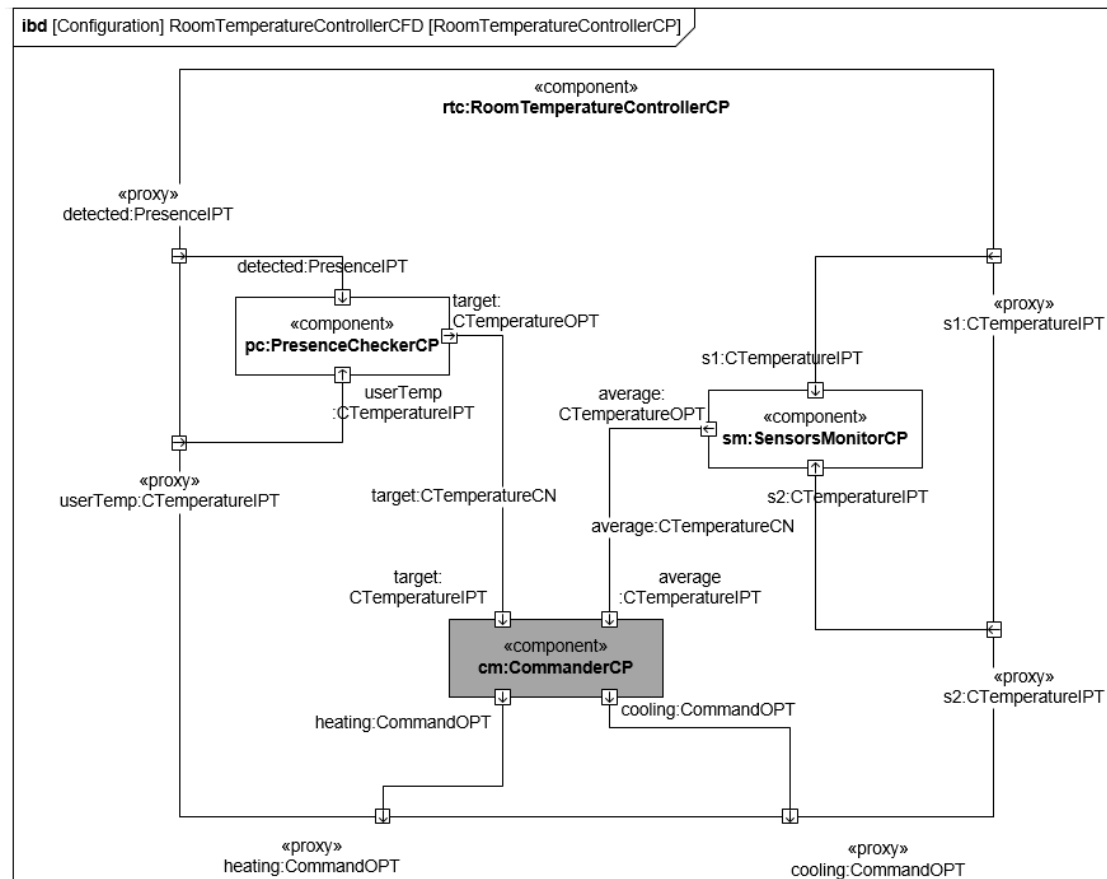


Example of architectural execution

Executing an architecture

Execution semantics

■ Execution step 8 (5th in inside 2nd level)

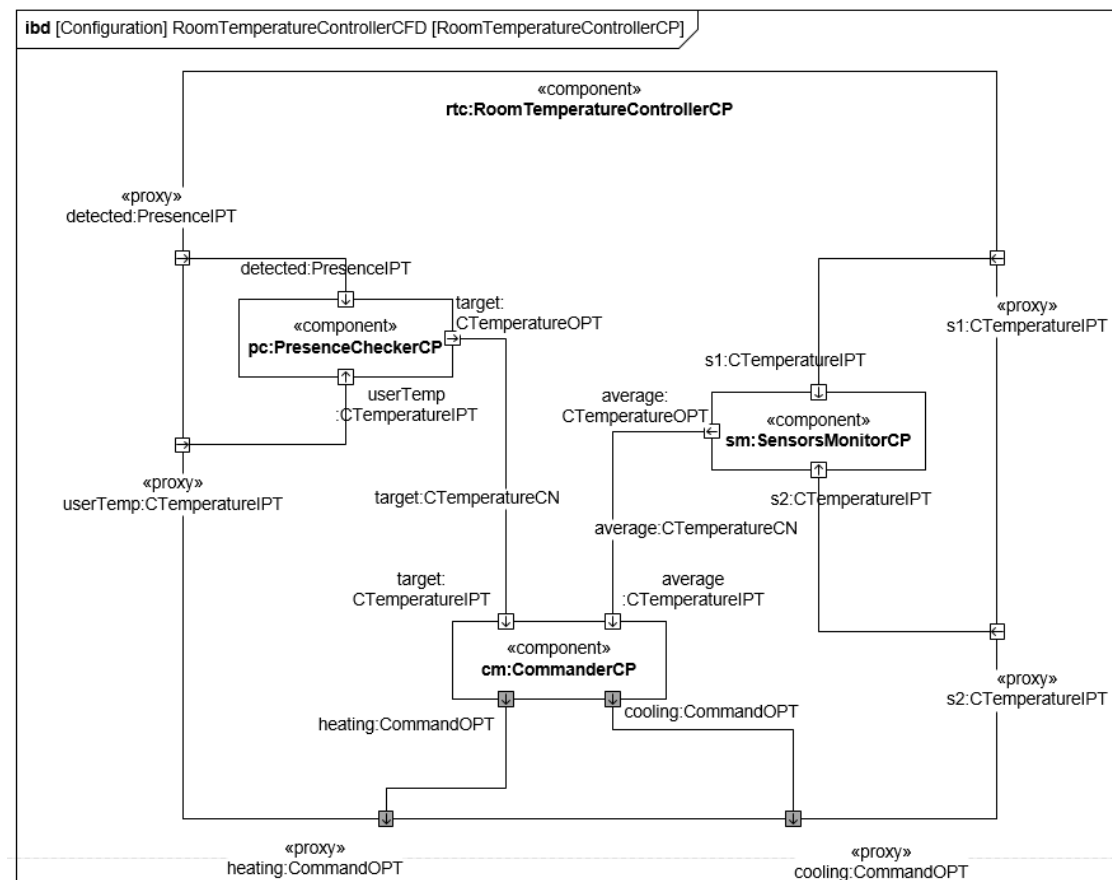


Example of architectural execution

Executing an architecture

Execution semantics

■ Execution step 9 (6th in inside 2nd level)

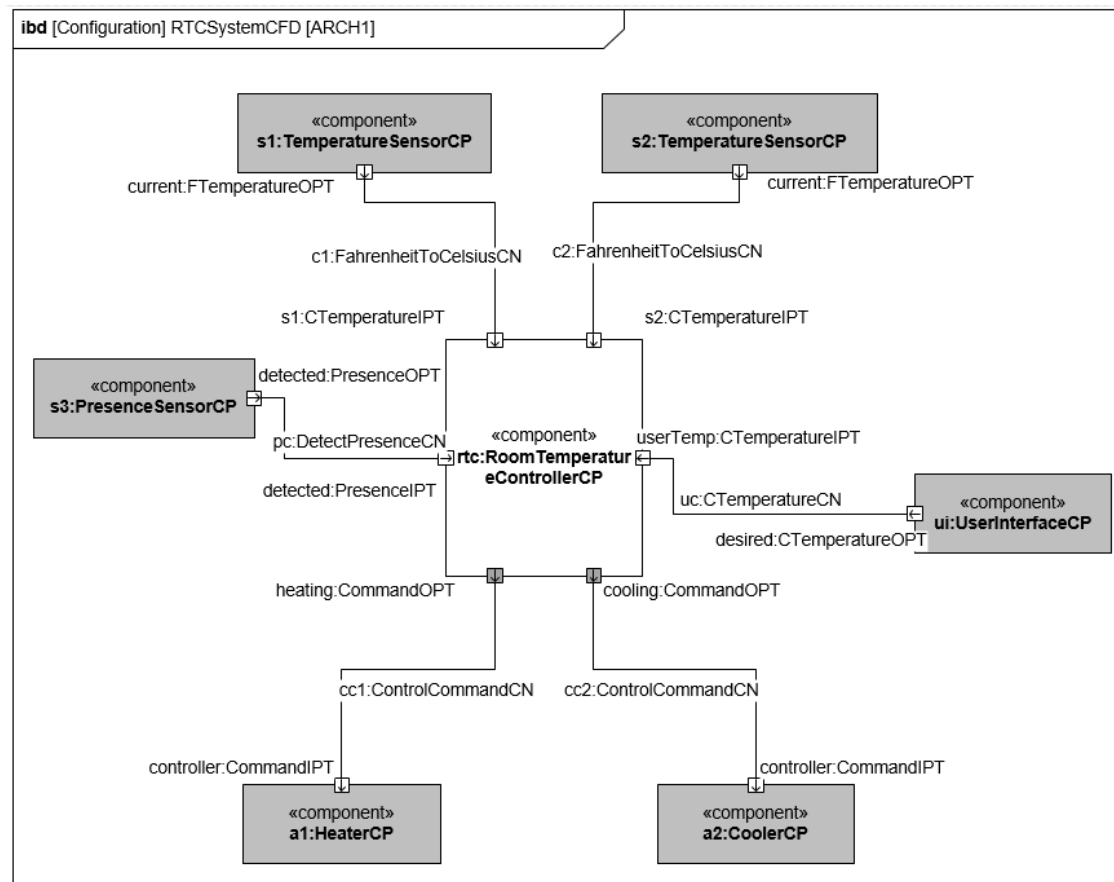


Example of architectural execution

Executing an architecture

Execution semantics

- Execution step 10 (coming back to the 1st level)

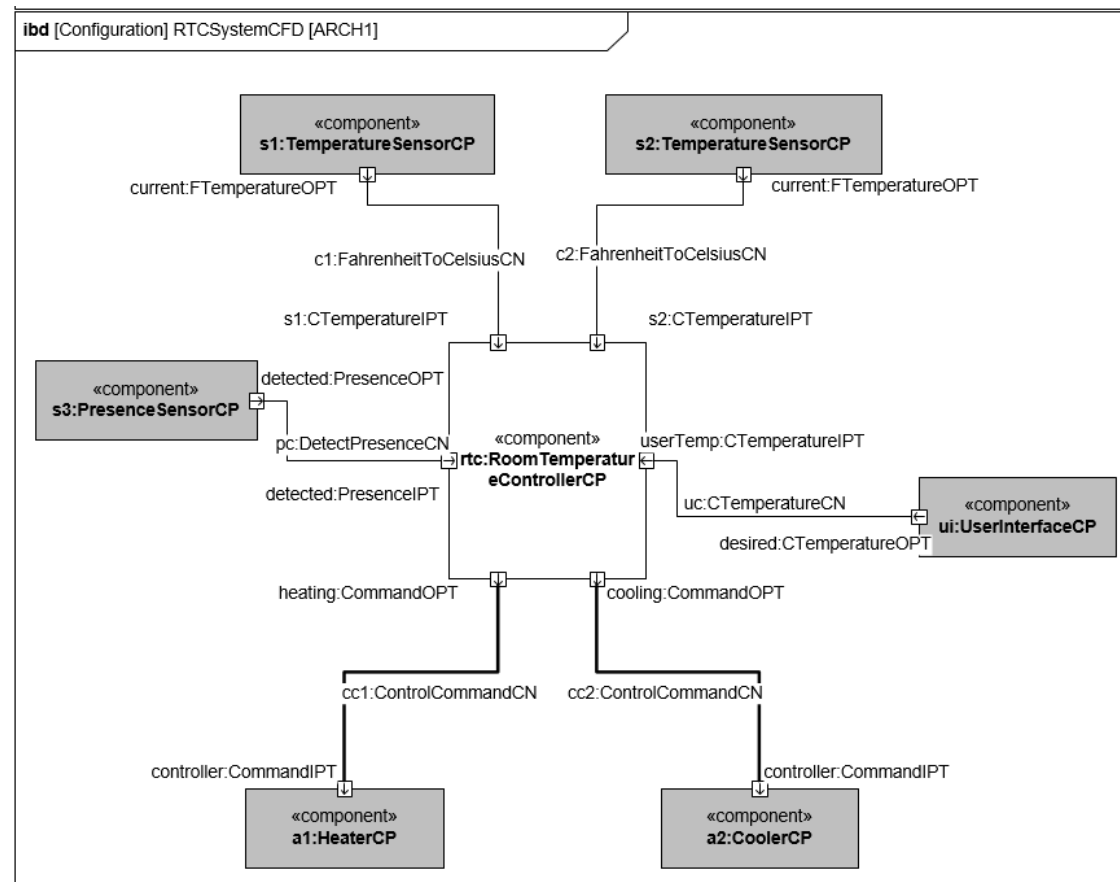


Example of architectural execution

Executing an architecture

Execution semantics

■ Execution step 11 (1st level)

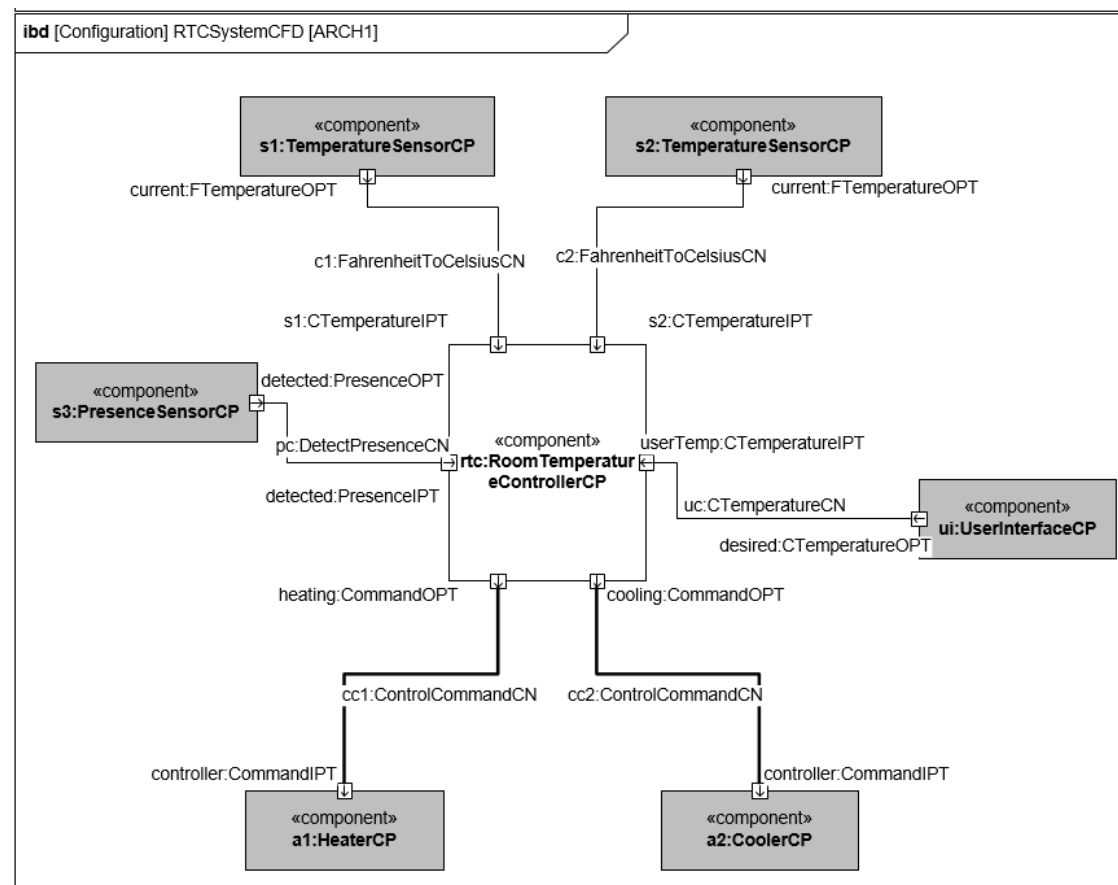


Example of architectural execution

Executing an architecture

Execution semantics

■ Execution step 12 (1st level)

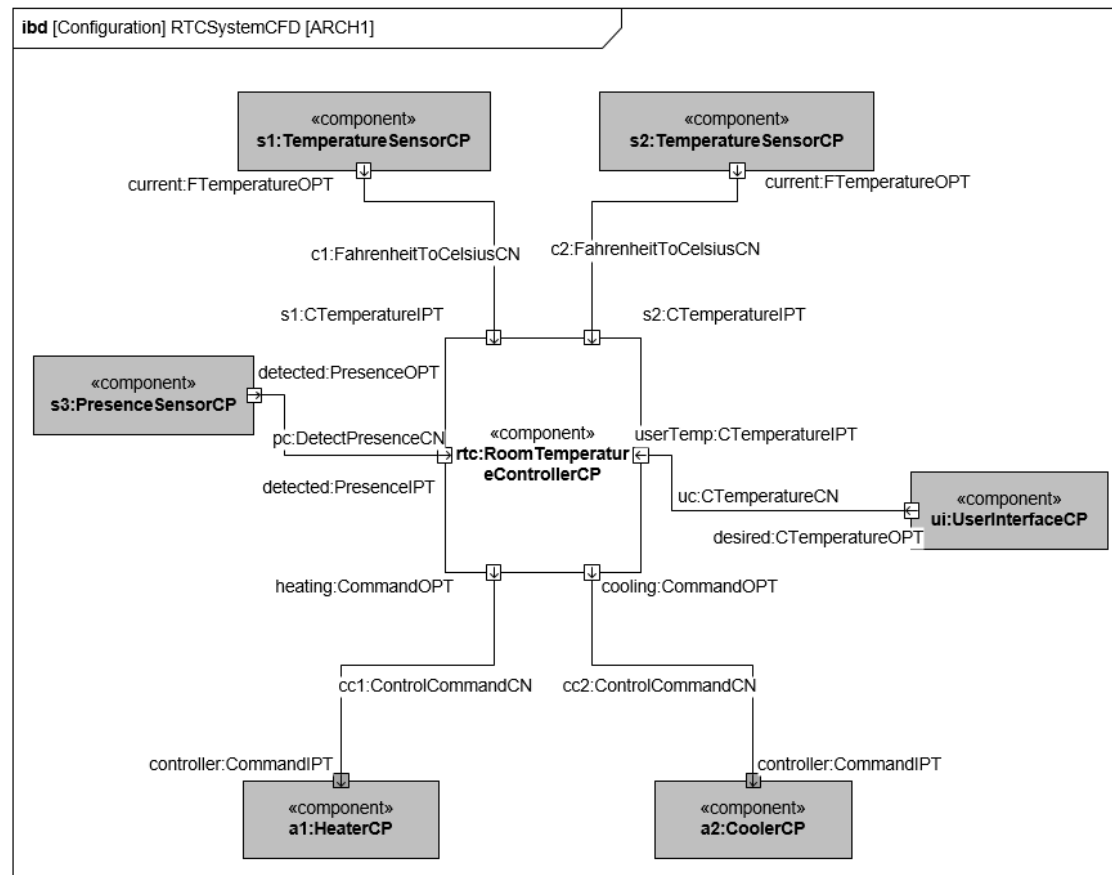


Example of architectural execution

Executing an architecture

Execution semantics

■ Execution step 13 (1st level)



Summary

- In this chapter, you learned:
 - how an architecture execution is expressed with SysADL
 - the operational semantics of the SysADL constructs
 - a scenario of an architecture execution of our running example

For further reading

- Concrete Syntax For A UML Action Language: Action Language For Foundational UML™ (ALF™).
<http://www.omg.org/spec/ALF/>
- Semantics Of A Foundational Subset For Executable UML Models (FUML™). <http://www.omg.org/spec/FUML/>
- Precise Semantics Of UML Composite Structures™ (PSCS™).
<http://www.omg.org/spec/PSCS/1.0/>