

Software Architecture in Action

Flavio Oquendo, Jair C Leite, Thais Batista



Chapter 4

Specifying the structure of software architectures

Learning outcomes of this chapter

- You will learn:
 - the architectural constructs to express structure: *components*, *connectors*, and *configurations*;
 - the auxiliary concepts: *port* for defining interfaces of components and connectors, and *value type* for data manipulation; and
 - the diagrams to represent these constructs.

+ The structure of this chapter

- Introduction
- Structural viewpoint and views
 - Structural viewpoint
 - Structural views
 - Component
 - Ports
 - Value types
 - Components with ports typed by value types
 - Connector
 - Configuration
 - Composite components
- Diagrams for structural views
 - Block definition diagrams
 - Internal block diagrams
- Describing the architecture from the structural viewpoint
- Summary
- For further reading

Introduction

Conceptual Overview – 1/2

- The *SysADL elements* used in the structural description of the architecture are:
 - **components** expressing the locus of computation,
 - **connectors** expressing the locus of communication, and
 - their **composition in configurations**.

Introduction

Conceptual Overview – 2/2

- From the structural viewpoint, the questions are:
 - what are **the concepts** required for specifying the structure of a software architecture?
 - what are **the SysADL constructs** provided by the structural viewpoint?
 - **how to apply** these constructs for describing different views of the architectural structure?
 - how **the diagrams** are used to express these views?



Structural Viewpoint

SysADL Viewpoints

Structural Viewpoint

Conceptual overview

- The ***Structural Viewpoint*** is concerned with the structure of the architecture, specifying the elements and how they are interconnected
- The Structural description includes the specification of the definition of software architecture elements and how they are connected together to interact to achieve the system functionality
 - the **definition of components, connectors, ports, values and configurations types**
 - **the use** of all these element to specify software architecture configurations



Structure

Structure

Conceptual overview

- The structure of a software architecture refers to **the way we assembly elements** to achieve the required system functionality
 - components, connectors, and configurations can have a structure
 - we can describe the structure of a software architecture or of a composite component
- An example of a software structure is the **definition of the elements that are parts of others**. For instance, the ports of a component, or a configuration of connectors and components that together form a composite component

Structural view

Conceptual Overview – 1/2

- A *structural view* shows a subset of the structural description of a software architecture including:
 - **the definitions** of components, connectors, and configurations (with associated ports and data) that are used in the definition of the architectural structure
 - how the defined components, connectors, and configurations **are used and composed** to define the architectural structure to achieve the required system functionalities and quality properties

Structural view

Conceptual Overview – 2/2

- For instance, in the specification of the architectural structure of the *RTC* System, we specify each component (the different sensors, the controller, and the actuators) and each connector linking a sensor to the controller, and the controller to an actuator.

Structural view diagrams

- Views are created to **organize the structural description of the architecture** according to the concerns of different stakeholders
- SysADL uses the following diagrams:
 - block definition diagrams (**bdd**), and
 - internal block diagrams (**ibd**)



Components

Components

Conceptual overview (1/3)

- A *software component* refers to a software element that realizes computations to achieve the required system functionality
 - it can be a **simple element** with a simple behavior, or a complex element representing a system as a whole
 - it interacts with the external world using *ports* (next slides)

Components

Conceptual overview (2/3)

- Software components can be **simple** or **composite**
 - **Simple components** perform simple computations using data in their ports
 - **Composite components** have other components in their internal structure
- Software components can be **internally located** or in the **boundary** of the environment
 - **Boundary components** are located at the interface with the environment
 - **Internal components** are located inside a composite component including the architecture itself

Components

Conceptual overview (3/3)

- As an example, the RTC System has
 - a **boundary component** that measures the temperature and provides its value in an output port.
 - a **simple component** that calculates the average temperature from several temperature providers.
 - a **composite component** that encompasses three simple components to control the temperature of a room.

Components

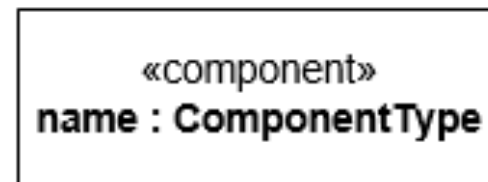
Use in SysADL

Component use

- We can use a component to represent a part of the system that perform a functional requirement
- When we use a component we should provide **its name** and **the name of its component type**
- We should **first define component types** and then create components of the defined type

SysADL notation

- We represent a component using a rectangle with a stereotype <<component>>, both for the component and its type
- We separate the name of each component and its type using a colon “:”



Component

Components localization

19

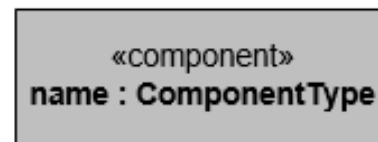
Use in SysADL

Component use

- A component may be located completely **inside** or **in the boundary** of the system architecture
- A component that is located in the boundary represents the system interface with the environment, encapsulating the mechanisms the system uses to send and receive data.

SysADL notation

- We represent a component located in the boundary using the same notation of the other components but we fill the box with grey color



Boundary component

Components

Use in SysADL - Example

Example Description

- In this example, we have
 - **two boundary components** named s1 and s2 both of the *TemperatureSensorCP* type
 - **one *sm* component** of the *SensorsMonitorCP* type

SysADL notation

«component»
s1: TemperatureSensorCP

«component»
s2: TemperatureSensorCP

«component»
sm: SensorsMonitorCP

Components

Definition in SysADL

Component definition

- Before using a component we need to **define its type**
- As in the previous example, we represent a component type using a `<<component>>` stereotype
- Component types are named
 - As a convention, we use a **CP suffix** to the name of a component type
- We use **the grey color to represent boundary components**

SysADL notation

- In this example we have the definition of the types:
 - *TemperatureSensorCP*
 - *SensorsMonitorCP*



«component»
TemperatureSensorCP

The diagram shows a rectangular box with a grey background. Inside the box, the text «component» is written in a smaller font above the text **TemperatureSensorCP**, which is in a larger, bold font.



«component»
SensorsMonitorCP

The diagram shows a rectangular box with a white background. Inside the box, the text «component» is written in a smaller font above the text **SensorsMonitorCP**, which is in a larger, bold font.



Ports

Ports

Conceptual overview

- A **port** is an interaction point between a component and other elements.
 - It represents **how the information flow** to a component (in ports) from another component (out ports)
 - A port **can be composed** by other ports – composite ports
- We use a *value type* to specify the kind of information that flows in the port.
 - Ports do not provide operations or services, only information
- An example is a port that provides the temperature of a *temperature sensor* component

Ports

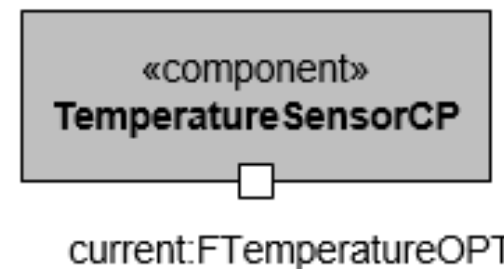
Use in SysADL – 1/3

Port use

- We use a port in a component to specify **what type of information is required** (in) or **provided** (out)
- When we use a port we should provide its name and the name of its port type (next slide)
- We should first define port types and then create ports of the defined type

SysADL notation

- A port is represented in a component as a small square in its border
- The temperature sensor below provides temperature values in Fahrenheit in its *current* port
- This port is an instance of the *FTemperatureOPT* port type (see later)



Ports

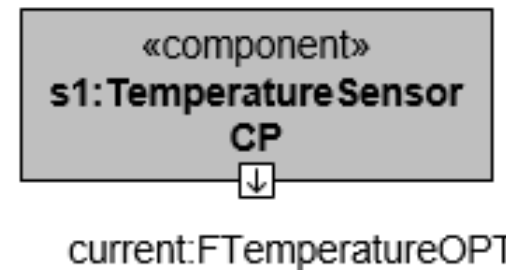
Use in SysADL – 2/3

Port use

- We can inform the flow of information (in or out).
- **Input ports** require information flowing into the component.
- **Output ports** provide information flowing out of the component

SysADL notation

- We can represent the information flow in SysADL using an arrow inside the port square indicating its direction.
- The temperature sensor in the last example has a port with temperature values flowing out of the current port.



Ports

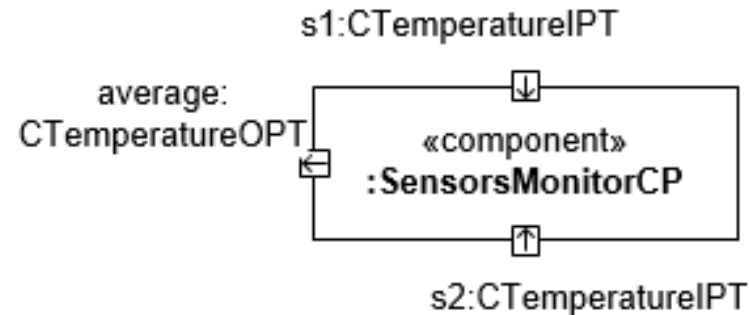
Use in SysADL – 3/3

Port use

- A component can have several ports of different types.
- All port types must be defined (see later)

SysADL notation

- The figure below illustrates a component with three ports of two different types (*CTemperatureOPT* e *CTemperatureIPT*)



Ports

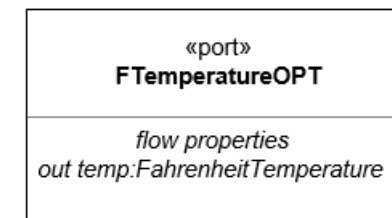
Definition in SysADL – 1/3

Port definition

- Before using a port as an element in a component, we need to define its type
- As a convention, we name port types using the following suffixes:
 - **OPT** to output ports
 - **IPT** to input ports
- The definition of a port type should include its flow properties in a specific compartment (next slide)

SysADL notation

- We define a port type using a `<<port>>` stereotype, its name, and its flow properties
- The *FTemperatureOPT* port type used in the previous example is now defined
 - It is an output port that provides temperature values in Fahrenheit
 - The *FahrenheitTemperature* value type must be defined (see later)



Ports

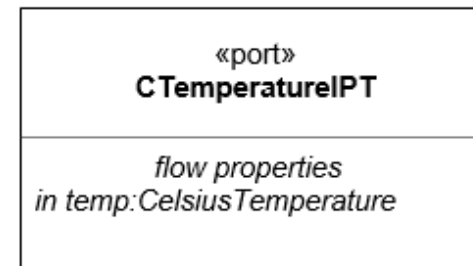
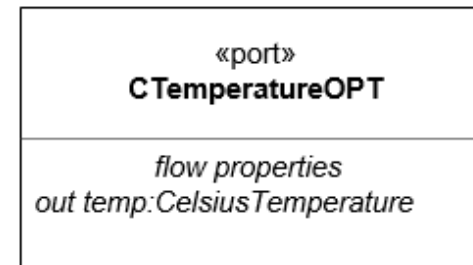
Definition in SysADL – 2/3

Flow properties

- When we define a flow property, we need to specify its direction:
 - ***in*** for information flowing into the port
 - ***out*** for information flowing out of the port
- A flow property should have a name and a value type

SysADL notation

- The *CTemperatureOPT* and *CTemperatureIPT* are both port types with information of the same value type flowing in different directions



Ports

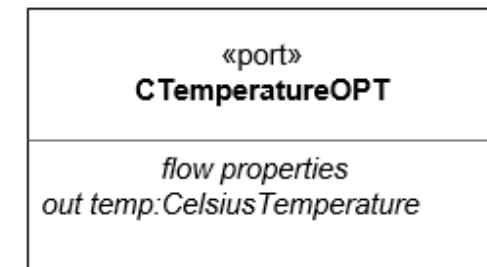
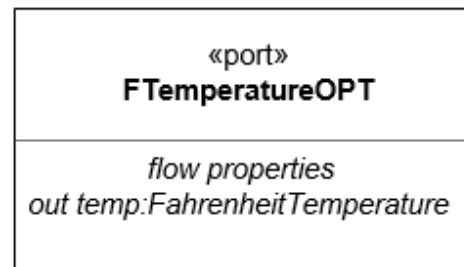
Definition in SysADL – 3/3

Flow properties

- When we define a flow property, we also need to specify **the value type of the flowing information**
- We can use a **primitive type** (boolean, real, integer, string) or a **user-defined value type**.

SysADL notation

- The *FTemperatureOPT* is a port type
 - instances of this type provide Fahrenheit temperature values
- The *CPTemperatureOPT* is a port type
 - instances of this type provide Celsius temperature values.



Composite Ports – 1/2

30

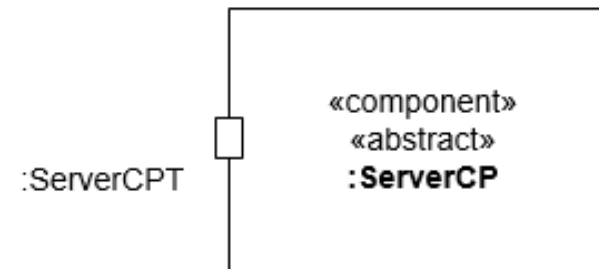
Use in SysADL

Port use

- A **composite port** can have several ports of different types in its structure
- We use a composite port as any simple port
- As a convention, we use a **CPT suffix** to the name of a composite port

SysADL notation

- The figure below illustrates a composite port



Composite Ports – 2/2

31

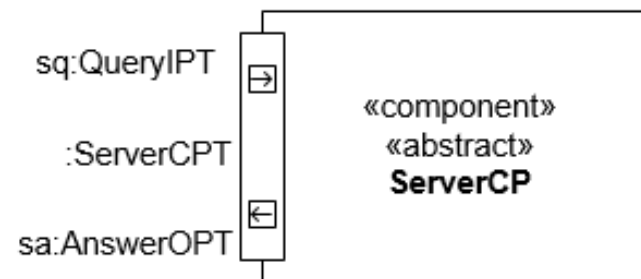
Use in SysADL

Port use

- A composite port can connect to a composite connector
 - the composite connector must be type compliant
 - the internal components must of the same type of the internal ports they connect with

SysADL notation

- The figure below illustrates the same composite port (ServerCPT) now with two ports of two different types (*QueryIPT* and *AnswerOPT*)



Composite Ports

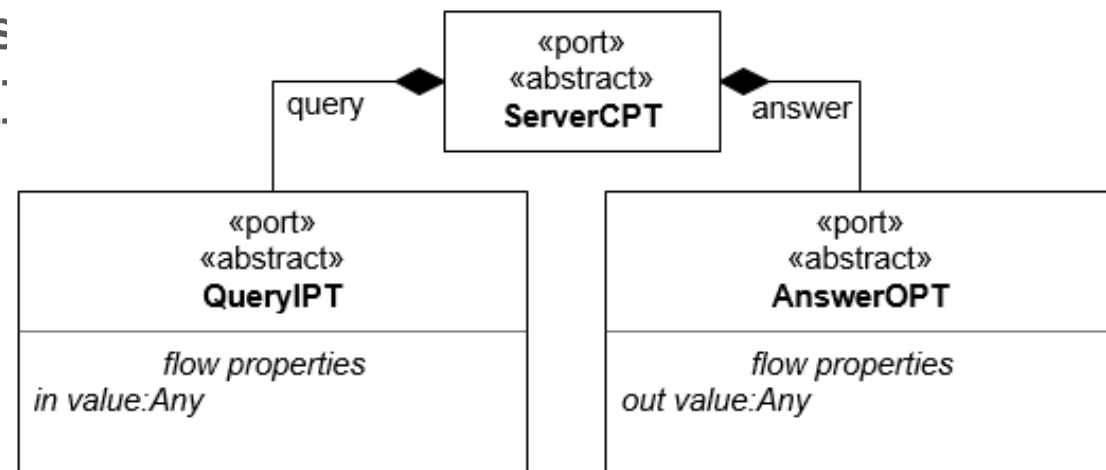
Definition in SysADL

Composite port definition

- We define a **composite port type** by specifying it as a composition of ports
- we need to specify the flow properties type of the internal ports

SysADL notation

- The *ServerPT* a port type is defined as a composition of two other ports: *QueryIPT* and *AnswerOPT*
- Each port can receive and send values of type *Any*



Conjugate Ports

Use in SysADL

33

Port use

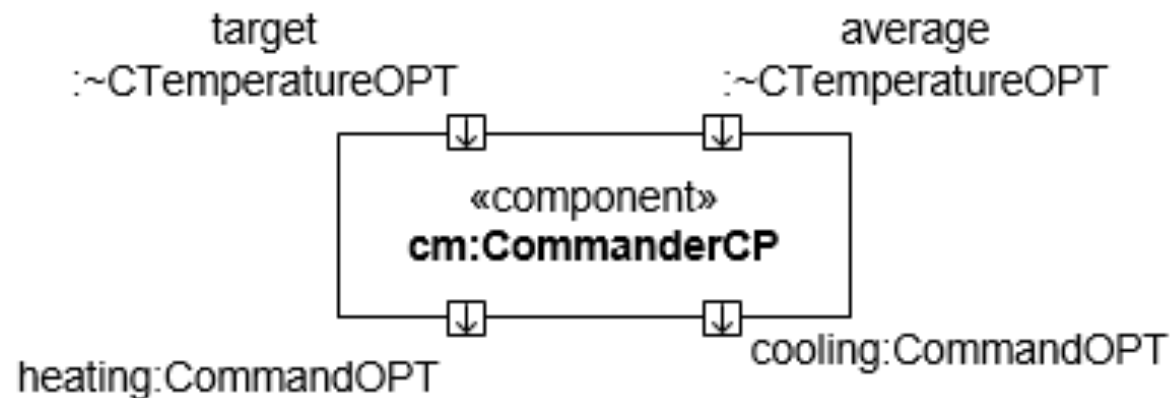
- A conjugate port is a **port of the same type of an original port but its flow port is the opposite.**
- Conjugate ports are indicated by a tilde '∼' before the type name
 - **Composite ports** can also be conjugated
- We use a conjugate port **to simplify the definition of ports** since we do not need to define it
- We also use a conjugate port when defining a connector (see later)

Conjugate Ports

SysADL notation

Port use

- In this new version of the *cm* component, we use two ports (*target* and *average*) both of the type of the conjugate *CTemperatureOPT*
- This mean that they are the reverse of the defined out port *CTemperatureOPT*
- However, we believe this can confuse the reader, so we prefer to define the *CTemperatureIPT* as in the previous example





Value types

Value types

Conceptual overview – 1/2

- Value types are **new types of information** to be used in the architecture description
- We typically represent information using primitive data types. However, sometimes, we need to use more abstract information that are meaningful for stakeholders
 - Value types may have a **dimension** or not
 - Value types with a dimension are simply called value types, and may have an **optional unit**

Value types

Conceptual overview – 2/2

- Value types without a dimension can be
 - **primitive types**, as Real, Integer, Boolean, String and enumerations
 - **data types**, which are structured, defined using other types
- *Temperature* is an example of a value type that we can define to represent the temperature values in Celsius or Fahrenheit

Value types

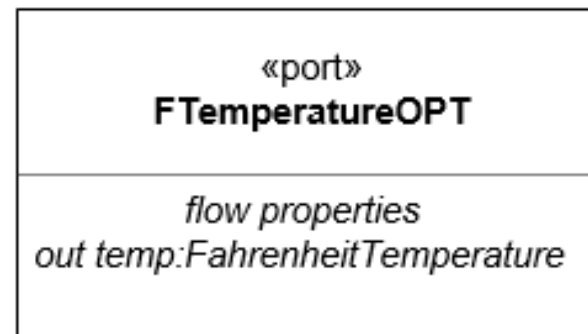
Use in SysADL

Value types use

- We can use the value type in the definition of:
 - flow properties of ports
 - flow properties of connectors (see later)

SysADL notation

- In the *FTemperatureOPT* port type used in the previous example illustrates the use of a value type in the output port that provides temperature values in Fahrenheit (the *FahrenheitTemperature* value type)



Value types

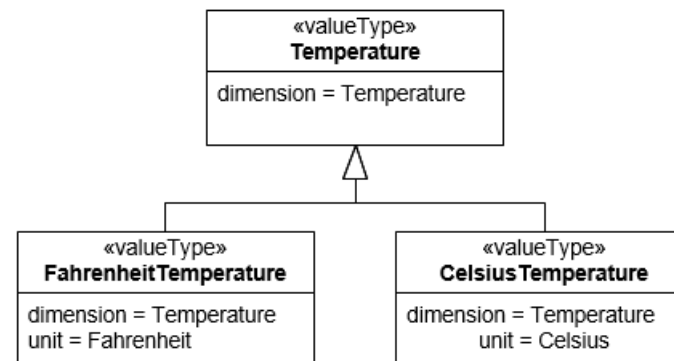
Definition in SysADL – 1/2

Value type

- A value type has a ***dimension*** and a ***unit***, which have types themselves
- ***dimension*** is a measurable extent of some kind such as temperature, force, power, mass
- ***unit*** is a unit of measurement of a dimension such as, for temperature could be Celsius or Fahrenheit, for power could be Watts

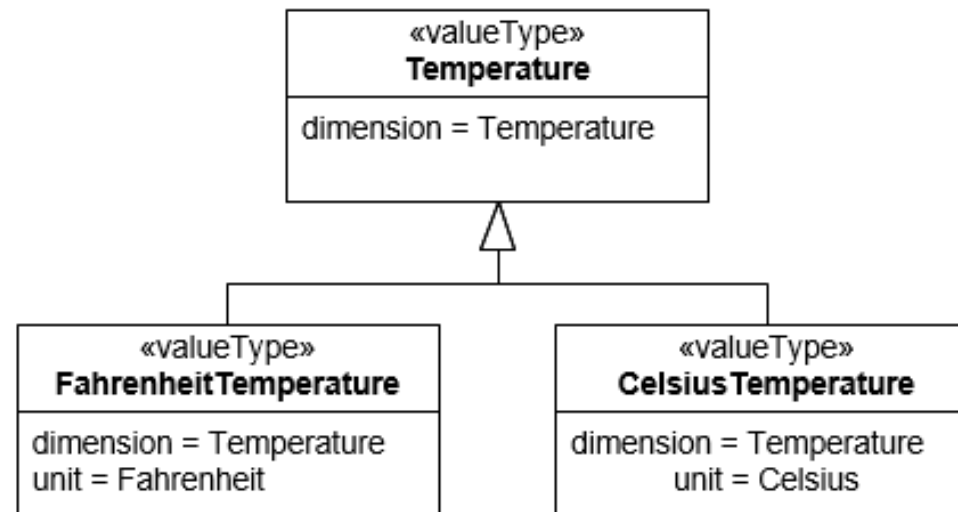
SysADL notation

- We define the type of information using the `<<valueType>>` stereotype
- The example below shows a definition of a *Temperature* supertype and two subtypes of specific units types to the same dimension (see next slide)



Value Types

Example figure



Value types

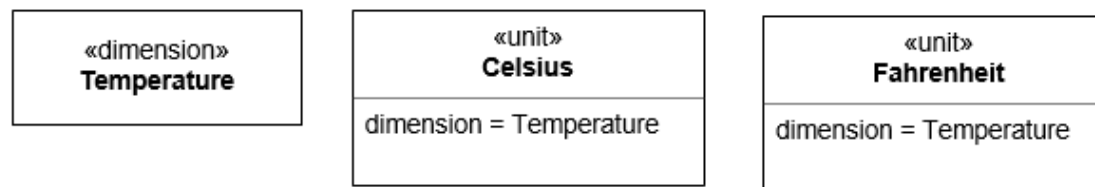
Definition in SysADL – 2/2

Dimensions and units

- We can define dimensions and units using the <<dimension>> and <<unit>> stereotypes, respectively
- The definition of an unit requires the associated dimension

SysADL notation

- The example below shows:
 - the definition of *Temperature* as a dimension
 - the definition of two units associated to the temperature dimension: *Celsius* and *Fahrenheit*



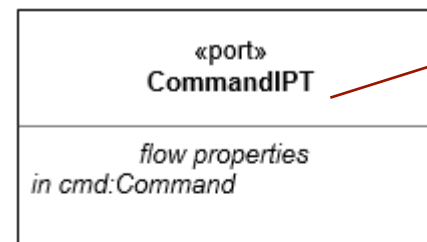
Primitive and data types

Use in SysADL

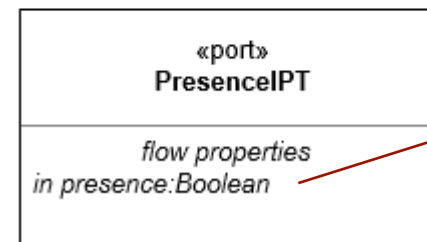
Primitive and data types

- Value types without a dimension can be
 - primitive types, as Real, Integer, Boolean, String and enumerations
 - data types, which are structured, defined using other types
- Primitive types are defined in SysADL and can be directly used
- Data types must be defined before used.

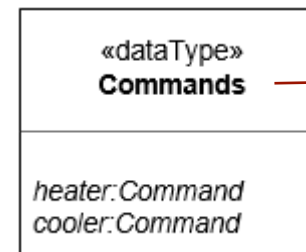
SysADL notation



Use of a
defined
data type



Use of a
primitive
type



Definition
of a data
type

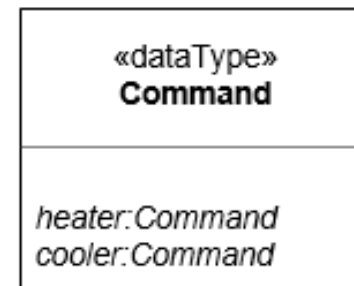
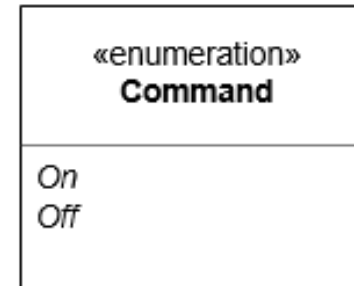
Primitive and data types

Definition in SysADL

Primitive and data types

- Before using information we need to define its type
- We define the type of information using the `<<dataType>>` stereotype

SysADL notation





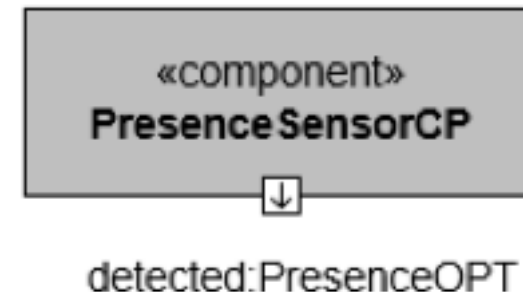
Components with ports typed by value types

Component type definitions in SysADL – 1/3

RTC System example

45

- The *Room Temperature Controller* System has the following component types with their respective ports:
 - A **presence sensor component type** (*PresenceSensorCP*) to detect if someone is in the room.
 - It has an **out port** – *detected* – that provides a value indicating a presence in the room.
 - The **port type** is *PresenceOPT*.

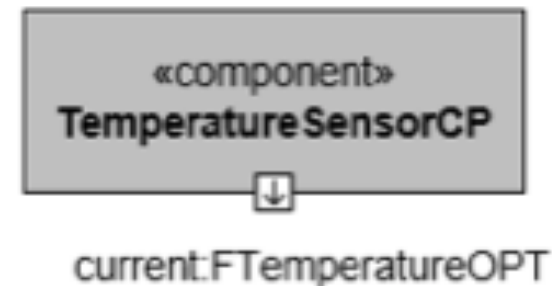


Component type definitions in SysADL – 1/3

RTC System example

46

- (Continued)
 - A **temperature sensor component type** (TemperatureSensorCP) to provide the current temperature value.
 - It has an **out port** – *current*– that provides the temperature value.
 - The **port type** is *FTemperatureOPT*

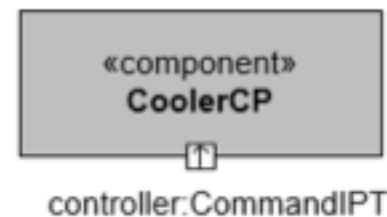
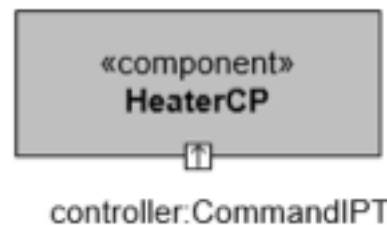


Component type definitions in SysADL – 2/3

47

RTC System example

- A **heater component type** (*HeaterCP*) to increase room temperature and the **cooler component type** (*CoolerCP*) to decrease it.
- They both have an ***in port*** – *controller*– that receives a command to the component.
- The port type is *CommandIPT*.

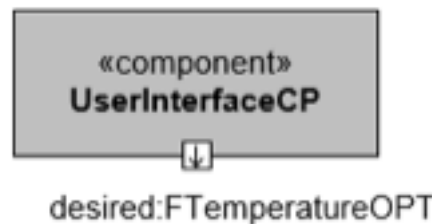


Component type definitions in SysADL – 2/3

48

RTC System example

- A **user interface component type** (*UserInterfaceCP*) to allow the user to provide the desired temperature value.
- It has an *out* port – *desired* – that sends the temperature value.
- The port type is *FTemperatureOPT*

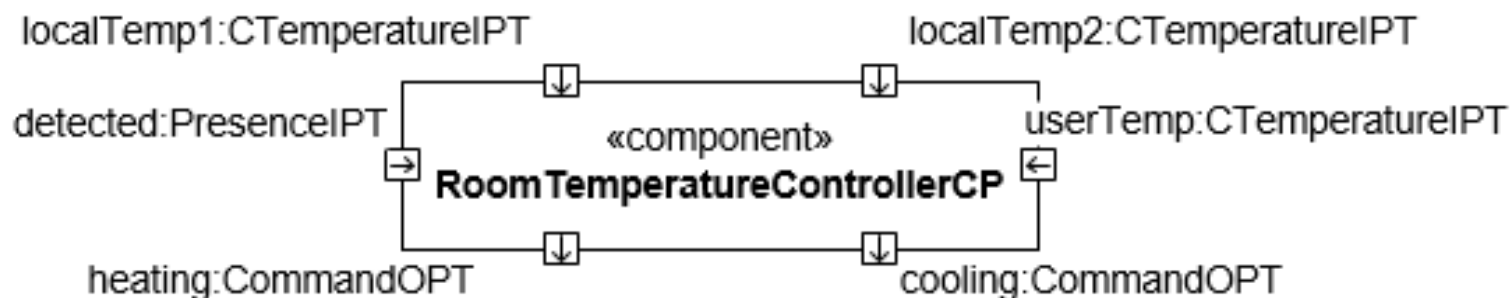


Component type definitions in SysADL – 3/3

49

RTC System

- A room temperature controller component type (*RoomTemperatureControllerCP*) to control the temperature by receiving temperature values and information about the presence of the user and by commanding the heater and cooler appropriately.
- The controller has the following ports:
 - **Two ports** to receive a Celsius temperature value.
 - **A port** to the user interface to receive the desired temperature value.
 - **A port** to receive the information about the presence of a person.
 - **Two ports** to provide the commands to the heater and the cooler.





Connectors

Connectors

Conceptual overview

- A *software connector* refers to software elements that represents the interaction between components
 - it is a simple element with a behavior
 - it links the ports of the connected components, allowing information to flow between them.
- An example is a connector that links the temperature port of a sensor to a temperature port of a controller

Connectors

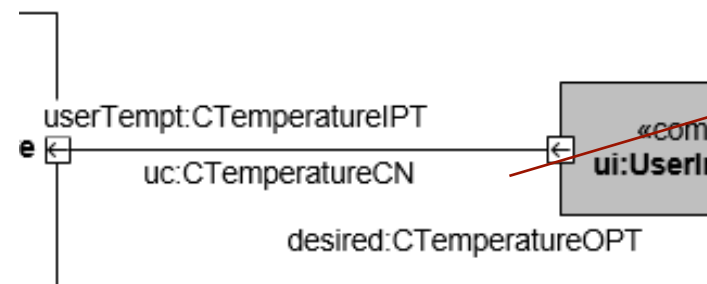
Use in SysADL

Connector use

- We can use a connector to represent **communication** between the components
- When we use a connector we should provide its name and the name of its connector type (see next slides)
 - You should first define connector types and then create connectors of the defined type
- We attach connectors to the ports of the involved components

SysADL Notation

- We represent a connector using a line
- We separate the name of each connector and its type using a colon “:”



Connector

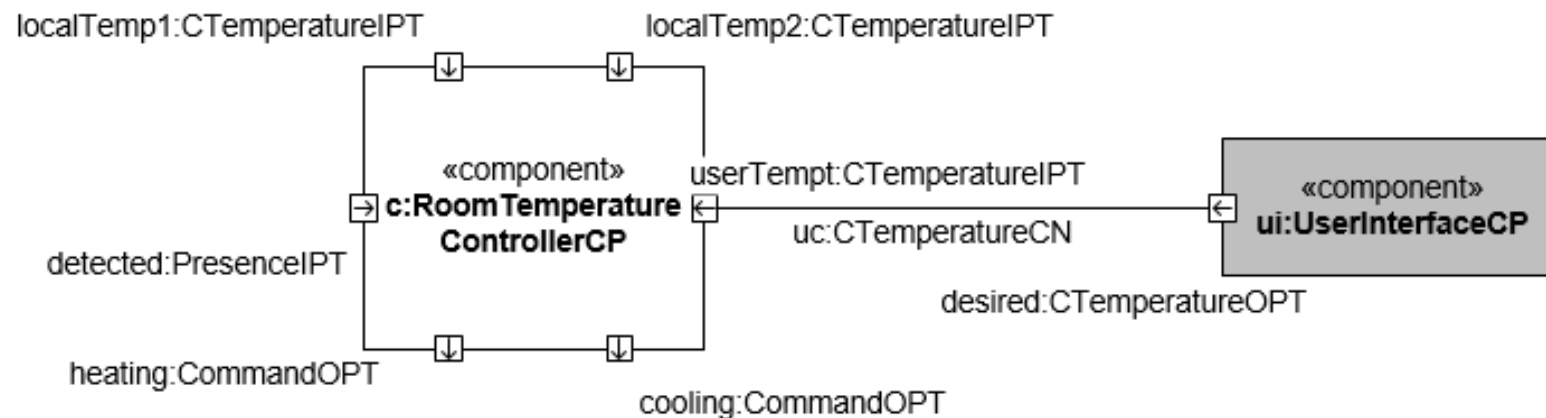
Connectors

Use in SysADL - Example

Example description

- In this example, we have
 - two components named *c* and *ui*
 - an *uc* connector of the *CTemperatureCN* type connects the components *c* and *ui*, using their ports *target* and *desired*
 - the information flows from *ui* to *c* components.

Example in SysADL



Connectors

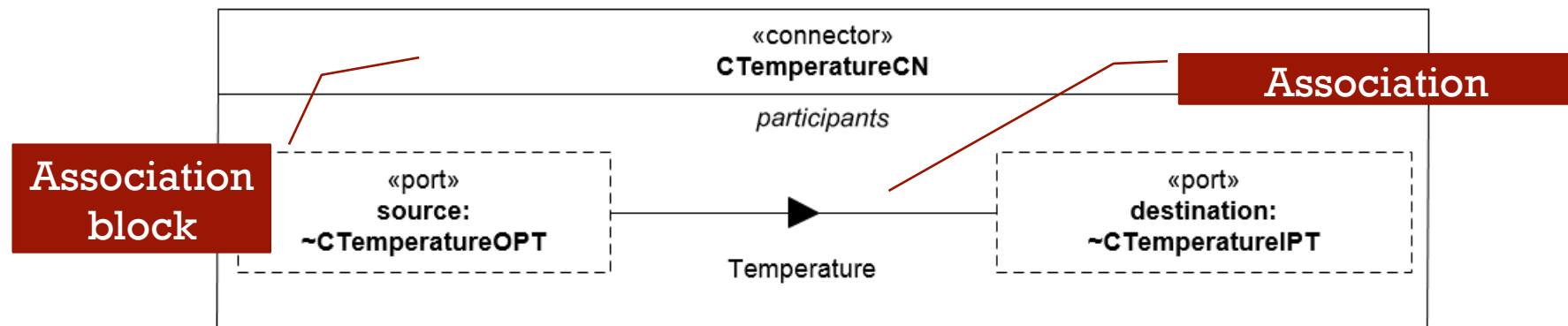
Definition in SysADL – 1/2

Connector Definition

- Before using a connector we need to define its type
- A connector type is an association block.
 - it defines an association between two other elements, the ports that are the participants of the association
 - we use the conjugation of the participants ports since in the connector they have the opposite role
 - the association indicates the two types of the participants ports and the value type of the information that flows through the connector

SysADL Notation

- We define a connector type using a <<connector>> stereotype.



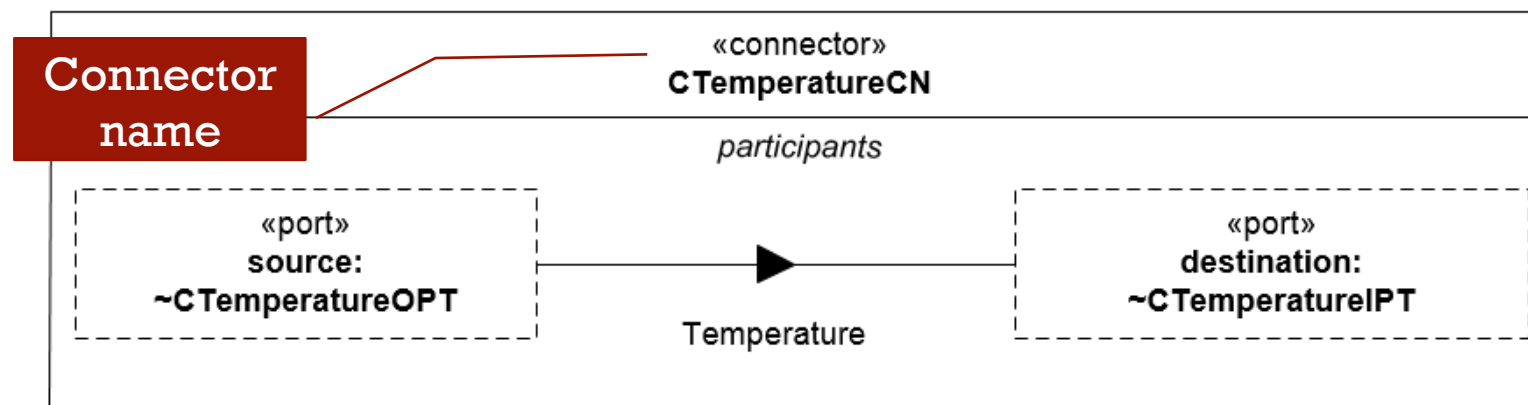
Connectors

Definition in SysADL – 2/2

Connector Definition - continued

- Connector types are named
 - as a convention, we use a **CN suffix** to the name of the connector type
 - we also use the names **source** and **destination** to indicate the role of each port in the information flow
- We define a connector type using a <<connector>> stereotype.

SysADL Notation



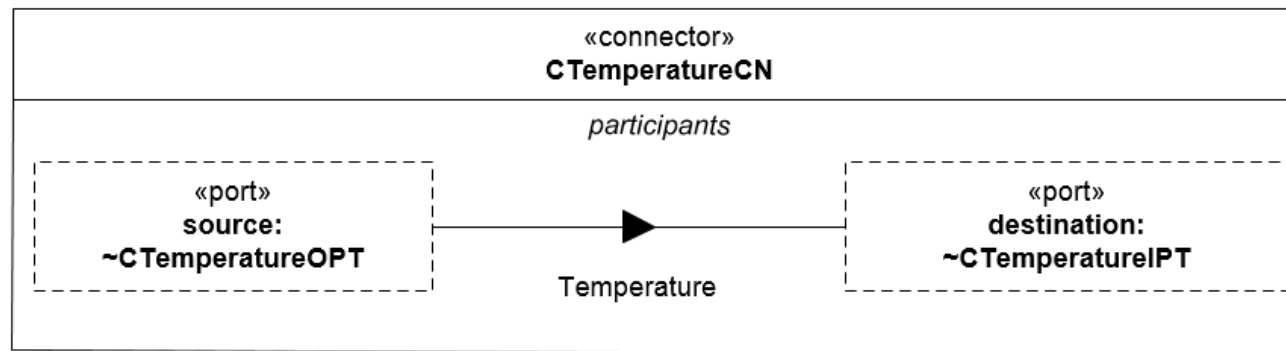
Connectors

Definition in SysADL - Example

Example Description

- The *CTemperatureCN* connector type defines the association between two ports types:
 - *CTemperatureOPT*, represented by its conjugation *~CTemperatureOPT*
 - *CTemperatureIPT*, represented by its conjugation *~CTemperatureIPT*
- The type of information flowing from the first port to the second is *Temperature*
 - Both ports have flow properties of type *CelsiusTemperature* that is a kind of *Temperature*

SysADL Notation



Composite Connectors

Use in SysADL

Example description

- A composite connector is used **to connect components that have composite ports**
- The connectors of composite connect must be defined and needs to be compatible with ports of a composite port they connect with
- In this example, we have a composite connector – *ClientServerCN* – that connects the composite ports – *ClientPT* and *ServerPT* (see in a former example)

Example in SysADL



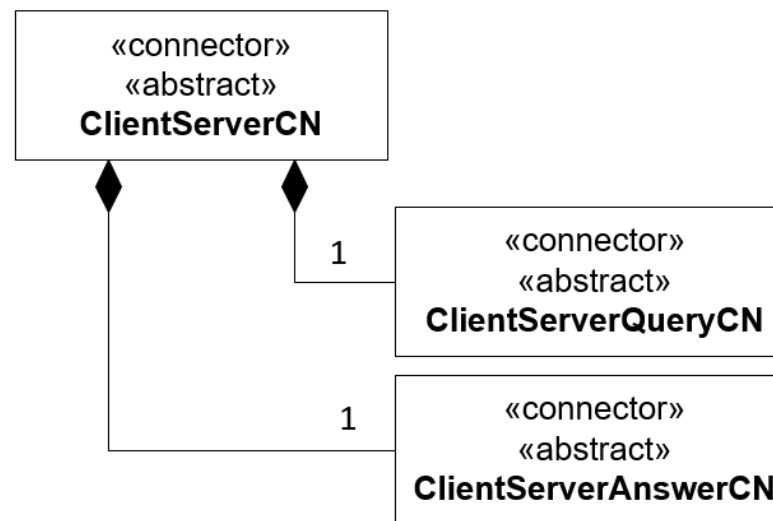
Composite Connectors

Definition in SysADL

Composite connectors example

- Before using a composite connector, we need to define it by specifying its constituent connectors
- We define each connector and use a composition to specify the composite connector (see next slide)

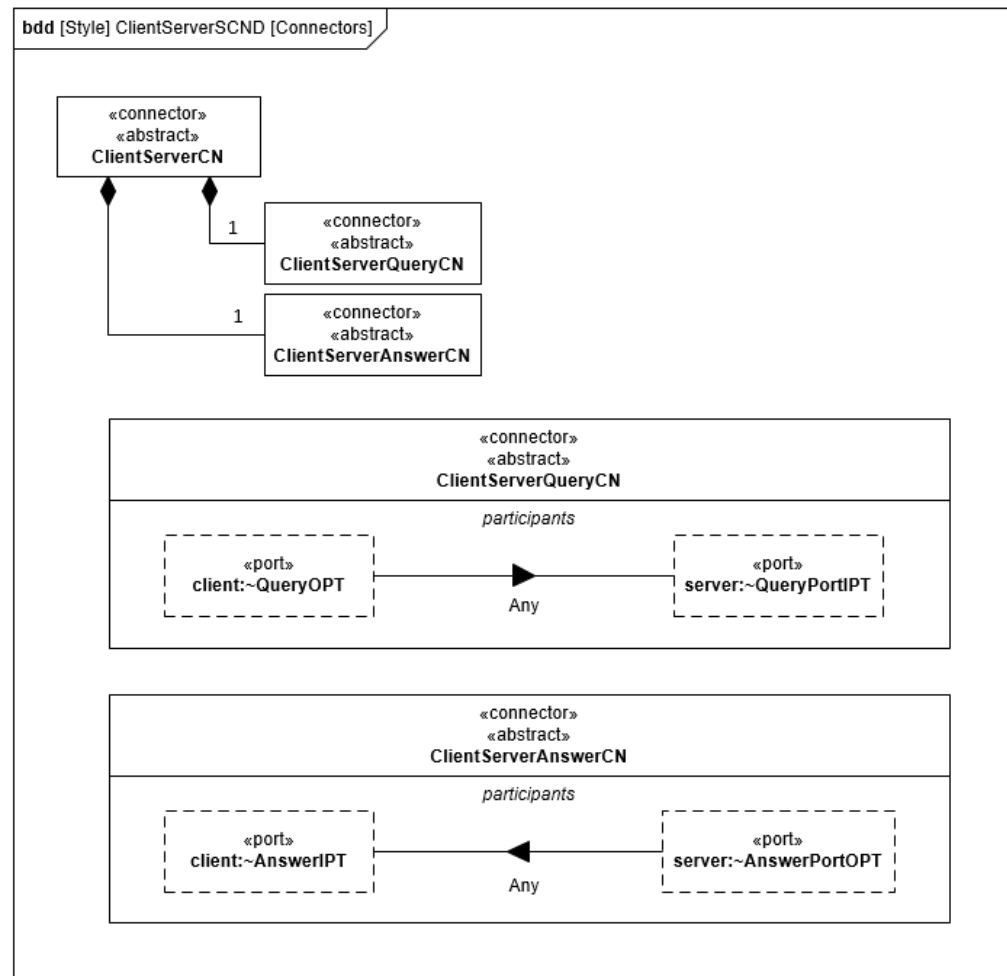
SysADL Notation



Composite Connectors

Definition in SysADL - Example

Composite connectors example – SysADL notation





Configuration

Configuration

Conceptual overview

- We use the concept of an architectural configuration to define how to connect components to
 - design composite components, or
 - design the overall software architecture of a system
- A configuration is the use of previously defined components and connectors to represent
 - the structure of the architecture
 - how components are connected using connectors to perform system functionalities

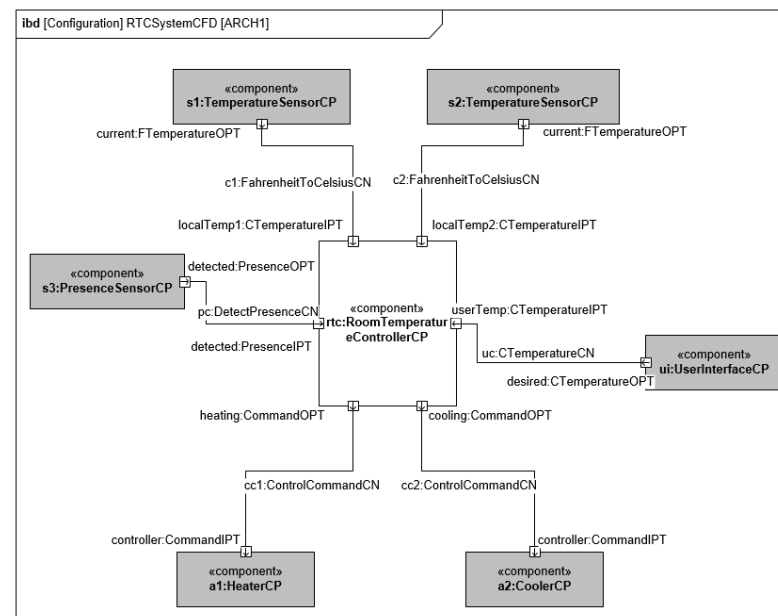
Configuration

Configuration definition using SysADL - ibd

Configuration definition

- We define a configuration in terms of the architectural elements previously presented: components and connectors
- We represent components and connectors with their types

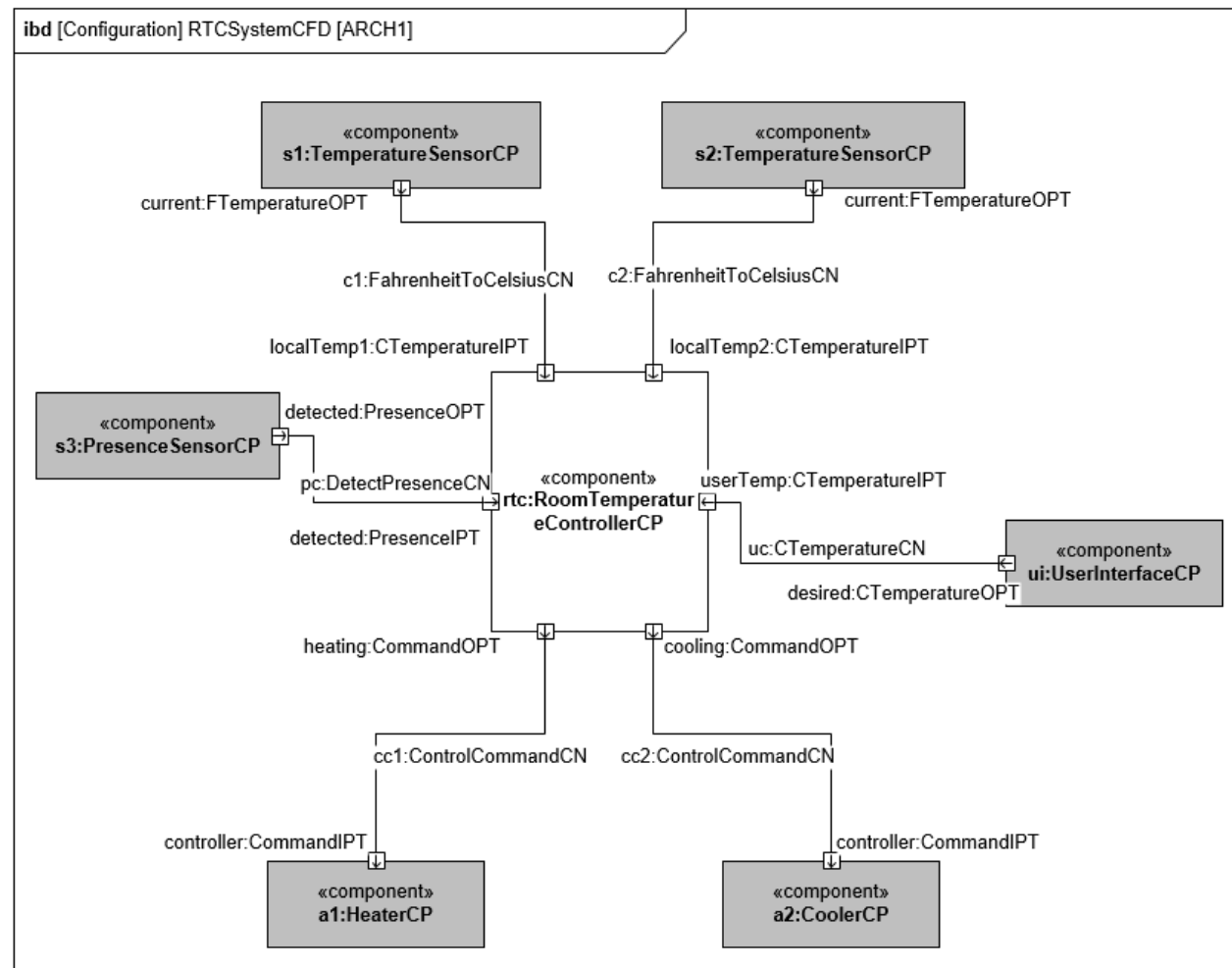
SysADL Notation



(larger in next slide)

Configuration

SysADL notation





Composite Components

Composite components

Conceptual overview

- A composite component is a component with an internal structure represented by a configuration
 - we define the configuration by specifying the components that are parts of the composition and by linking them using connectors
- We can define an internal configuration of a composite component using an ibd
- As an example, we have the configuration of the *RoomTemperatureControllerCF* composite component that contains three components and two connectors (next slides)

Composite components

Configuration example in SysADL

Example description

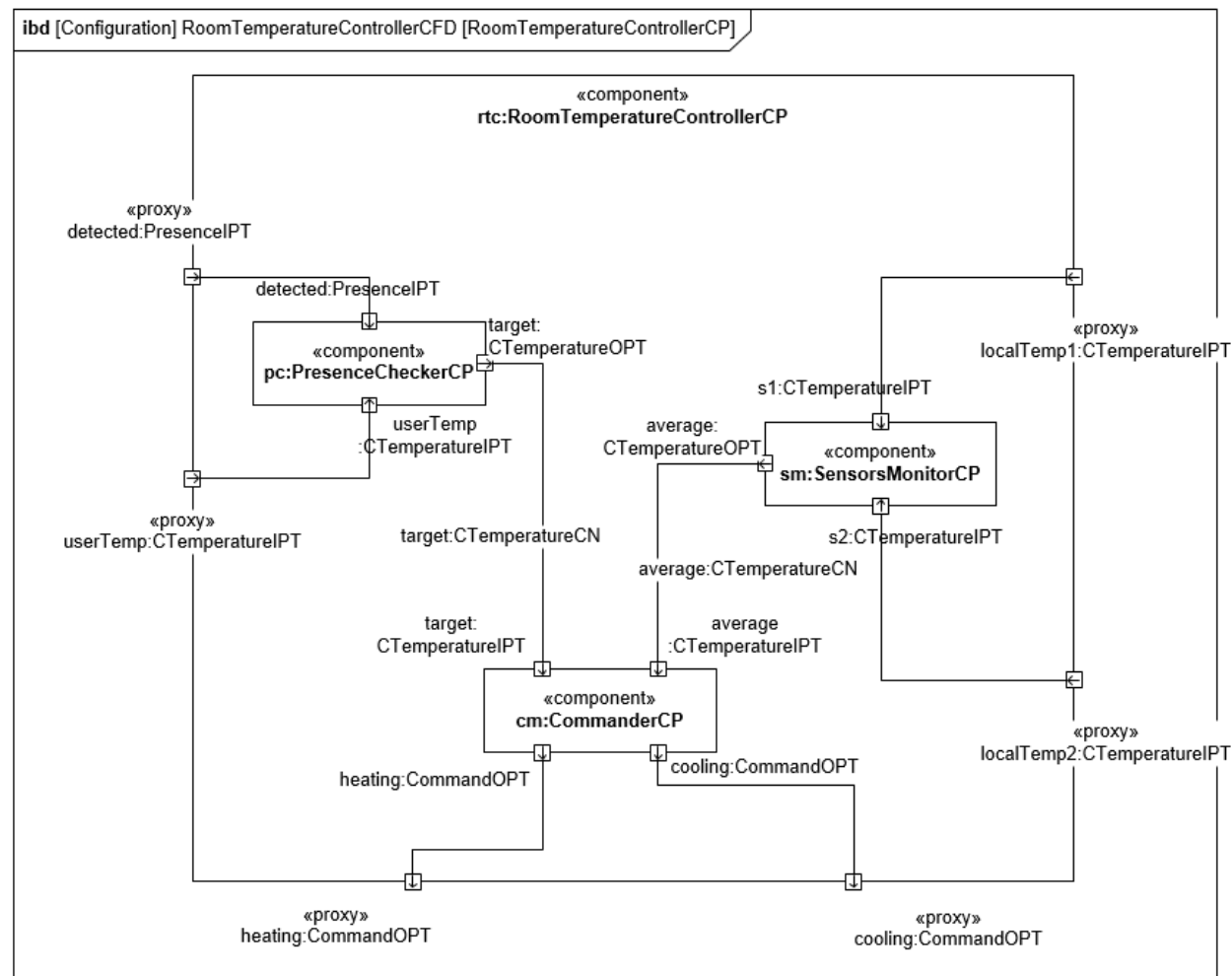
- The *RoomTemperatureControllerCF* composite component that contains three components and two connectors (next slide)
 - pc: a *PresenceCheckerCP* component type
 - sm: a *SensorsMonitorCP* component type
 - cm: a *CommanderCP* component type
 - *target* and *average* connectors: both of *CTemperatureCN* connector type

SysADL notation

- See next slide

Composite components

SysADL notation



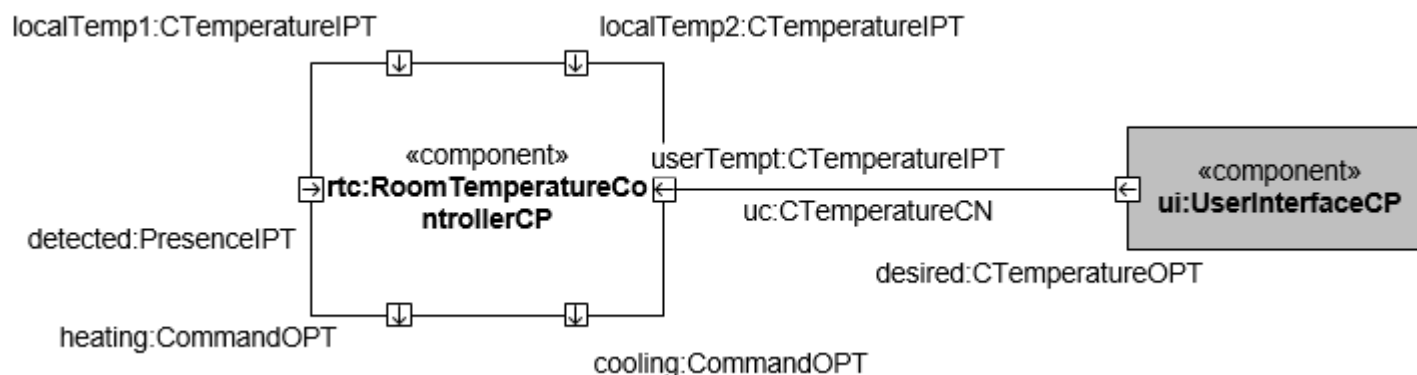
Composite component

Use in SysADL

Composite component use

- We use a composite component in the same way of a component
 - we should provide its name and the name of its previously defined component type
- The *rtc* component in the RTC system is a *RTCRoomTemperatureControllerCP* composite component type (previously defined)

SysADL notation



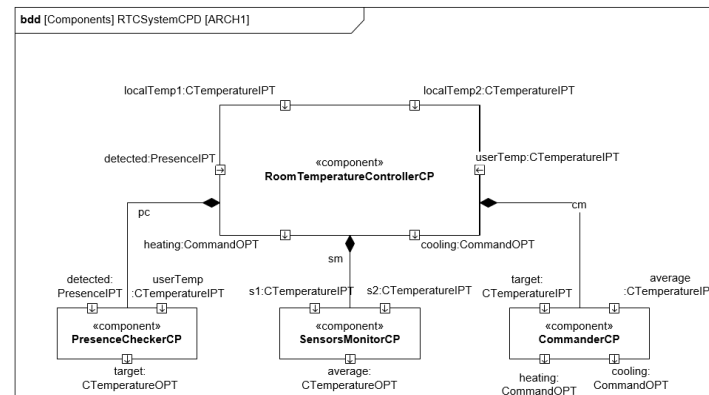
Composite component

Composite component definition in SysADL

Composite component definition

- We can define the composite component of the last example by using the compositions associations
- The *RoomTemperatureControllerCP* has three internal components (parts)
- The names used in the compositions are the same names of the components used in the configuration (*pc*, *sm*, and *cm*)

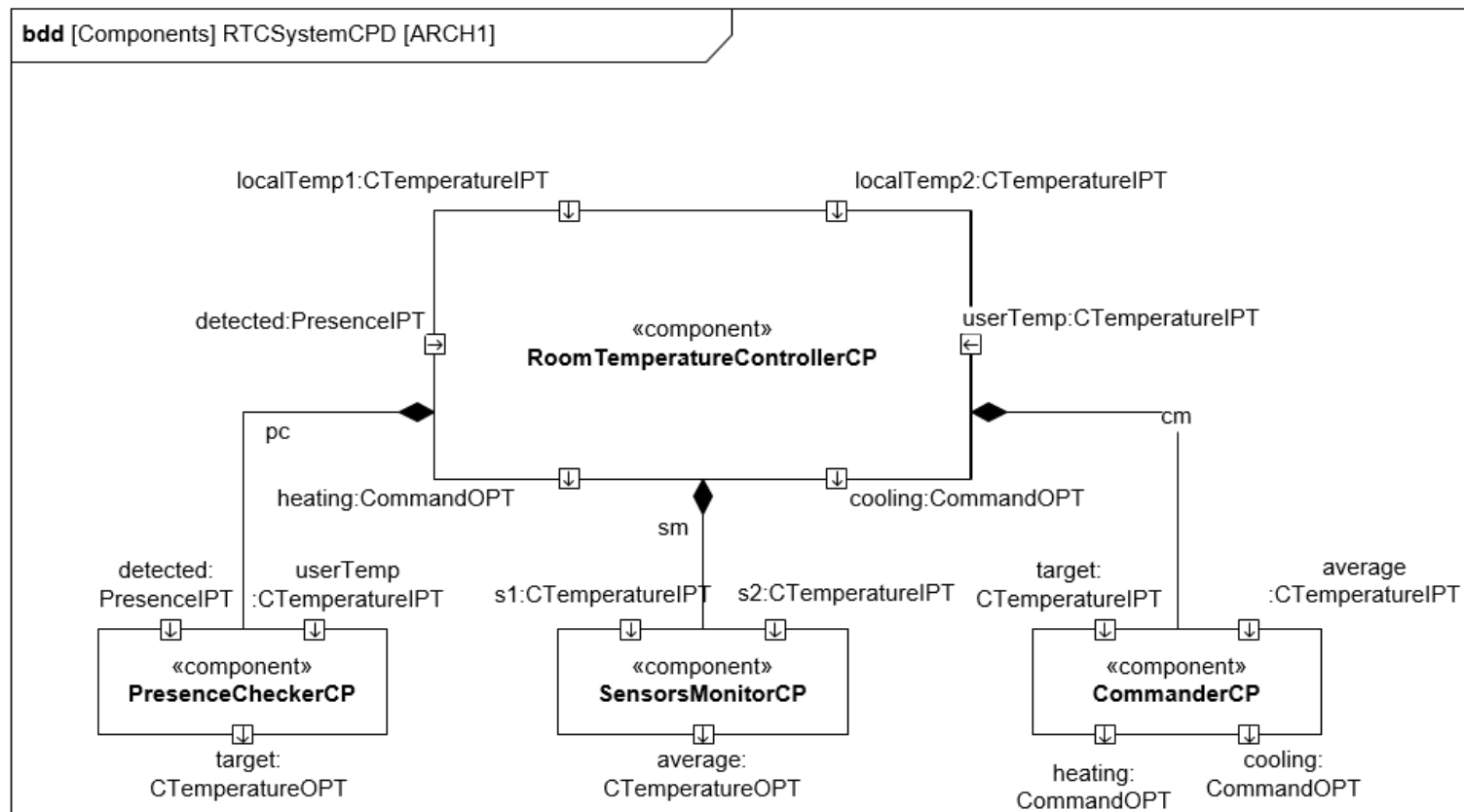
SysADL Notation



(larger in next slide)

Composite component

SysADL Notation



Composite component

Composite component configuration in SysADL

Composite component configuration definition

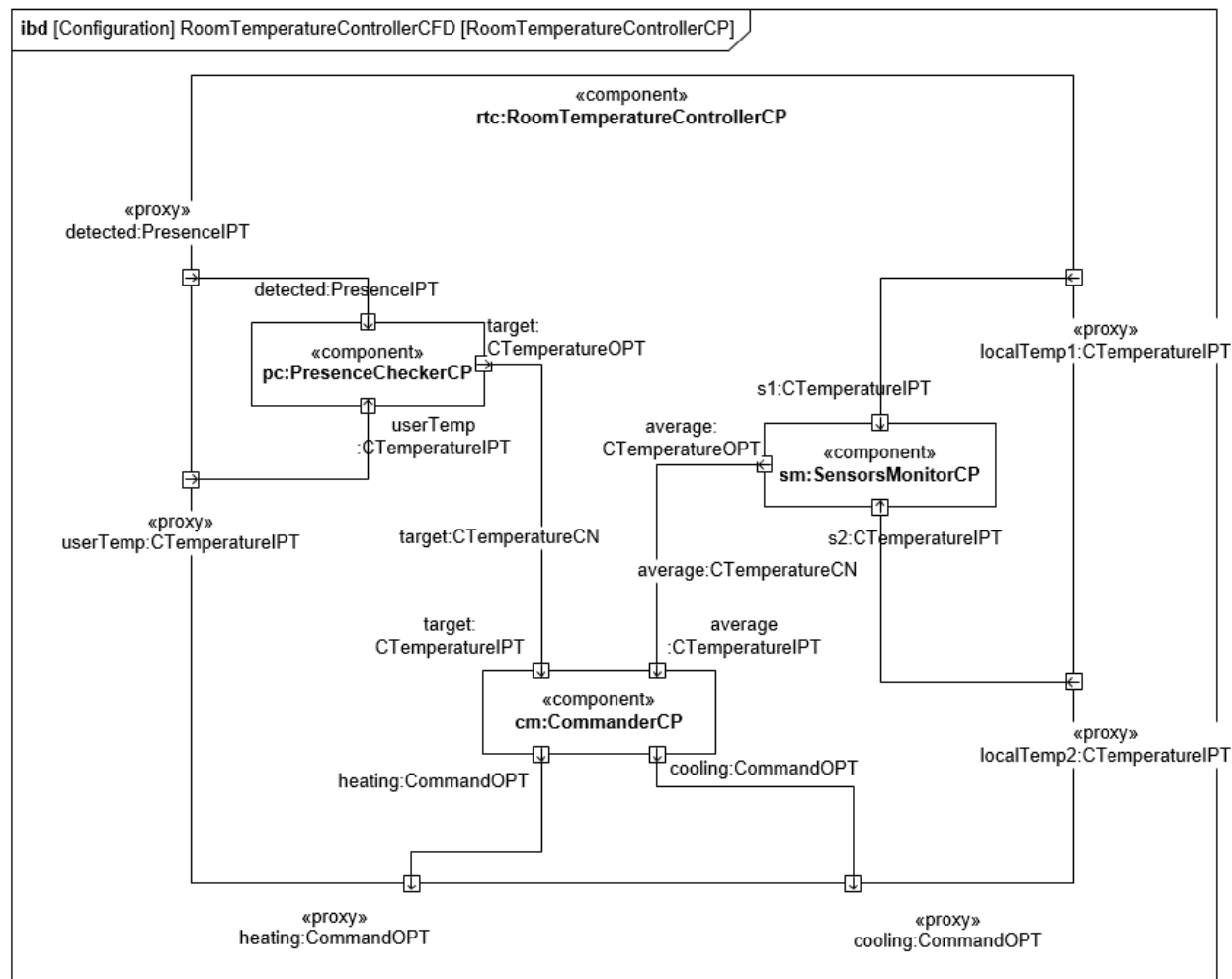
- We can define the configuration of a composite component using an *ibd*
 - We use three components: *pc*, *cm* and *sm*
 - Connector *target* connects *pc* to *cm*
 - Connector *average* connects *sm* to *cm*
- The ports of the internal components are not visible to the external world
- When a port needs to be visible, we link it to a proxy port via a binding connector
 - A proxy port specifies the features of the internal ports that are visible through external connectors
 - We use a binding connector to link an internal port to a proxy port

SysADL notation

- See next slide

Composite component

SysADL Notation





Diagrams for structural views

Diagrams for structural views

Overview

- Structural views are organized in terms of diagrams.
- The structural definitions of components, ports, value types and connectors are expressed using block definition diagrams (*bdd*).
- Configurations are defined with internal block diagrams (*ibd*).



Block Definition Diagrams

The block definition diagram (bdd)

76

Overview

- We define architecture types using *block definition diagrams* (bdd)
- It is important to note that in a modeling process we need to first define all elements before using it
- We define a **diagram for each type of element**
 - a bdd for **value types**
 - a bdd for **port types**
 - a bdd for **component types**
 - a bdd for **connector types**
- The following diagrams show the definitions of the types used in our example

Block definition diagram

BBD in SysADL - general

BDD header conventions

- We define a bdd specifying its **header** and its **contents**
 - in the the header, we inform the **diagram kind**, the **architectural model element kind**, the **name of the diagram**, the **name of an element**, and the **name of the architecture or specific component**
 - in the content, we define the elements

SysADL Notation

```
bdd [Components] RTCSysmCPD [ARCH1]
```

Block definition diagram

BBD in SysADL - header

BDD header conventions

- As a convention, in the the header we use the following elements:
 - **bdd** – indicates that the diagram is a block definition diagram
 - **[Model Element Type]** – indicates what kind of element the diagram specifies – optional since the elements have names
 - **Diagram name** – used to uniquely identify the diagram
 - As a convention we include a suffix to the name (see next slide)
 - **[Element]** – used to identify the architecture or a specific element

SysADL Notation

```
bdd [Components] RTCSysmCPD [ARCH1]
```

Block definition diagram

BBD in SysADL - names

BDD names conventions

- As a convention, we use the following suffixes in the diagram name
 - VLD – for value types diagrams
 - DUD – for dimensions and units diagrams
 - CPD – for component types diagrams
 - CND – for connectors types diagrams
 - PTD – for port types diagrams

SysADL notation

bdd [ValueTypes] RTCSysVLD

The diagram is a rectangular box with a diagonal line on the right side, representing a SysADL notation for a ValueTypes diagram.

bdd [Connectors] RTCSysCND

The diagram is a rectangular box with a diagonal line on the right side, representing a SysADL notation for a Connectors diagram.



Internal Block Diagram

SysADL Diagrams

The internal block diagram (ibd)

Overview

- We define architecture configurations using *internal block diagrams (ibd)*
- The ibd contains **instances of the defined architectural elements** linked together to express the software structure
- The following diagrams show the definitions of configurations used in our example

Internal block diagram

IBD in SysADL

IBD header conventions

- We define an ibd by specifying its header and its contents
- As a convention, in the header we use the following elements:
 - **ibd** – the diagram kind
 - **[Configuration]** – to inform the diagram represents a configuration.
 - the **name of the diagram**. As a convention, we use a CFD suffix to the name of the configuration.
 - **[Element]** – the name of the element the diagram depicts a configuration: composite component or software architecture

SysADL Notation

ibd [Configuration] RTCSysCFD [ARCH1]

ibd [Configuration] RoomTemperatureControllerCFD [RoomTemperatureControllerCP]

+ Describing
the
architecture
from the
structural
viewpoint

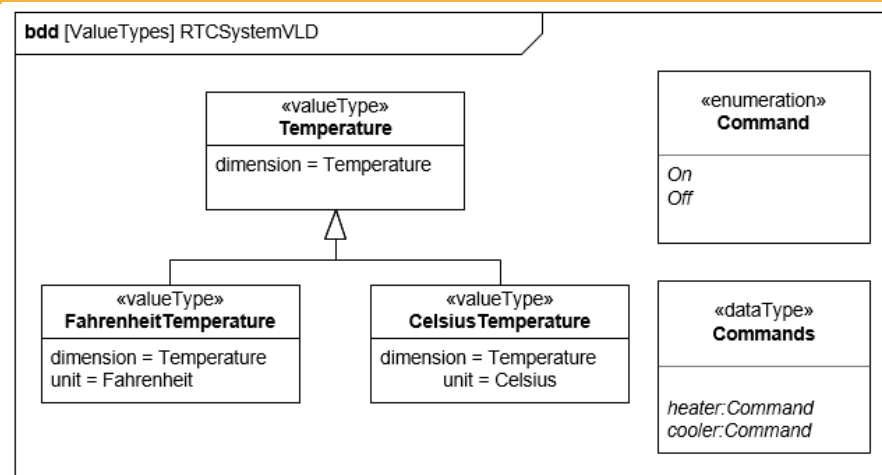
The RTC System

Value types of the RTC architecture description

Value types in RTC System

- We define the following value types for the RTC System
 - the *Temperature* value type with two specializations: *FahrenheitTemperature* and *CelsiusTemperature*
- We include **data types definitions and enumerations definitions** in the value type BDD.
 - the *Command* value type that is an enumeration of “on” and “off”
 - Commands define a data type that has two value types

SysADL Notation

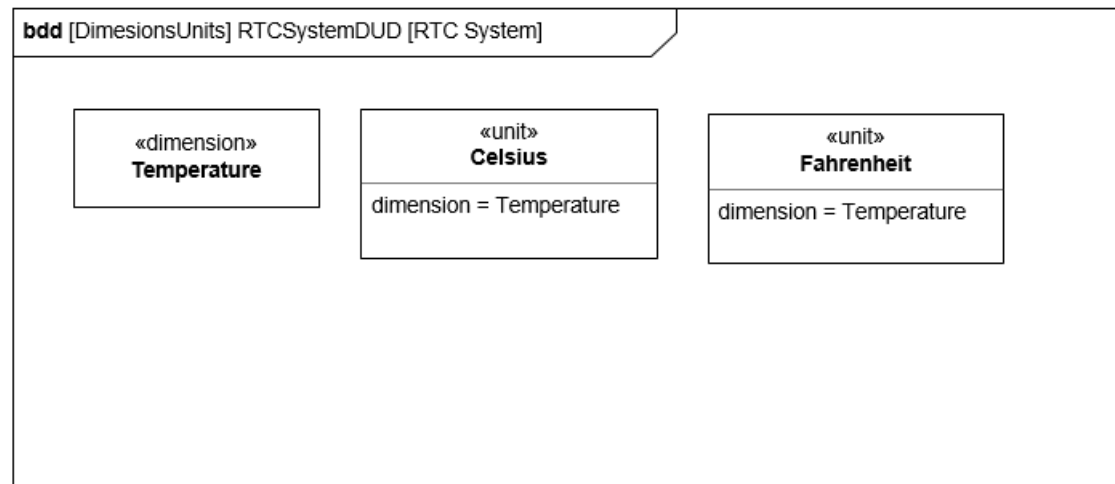


Dimensions and units types definitions of the RTC

Dimensions and units types in RTC System

- We define the following dimension type for the RTC System
 - Temperature
- We define two units associated to the *Temperature* dimension
 - Celsius and Fahrenheit.

SysADL Notation



Port types definitions

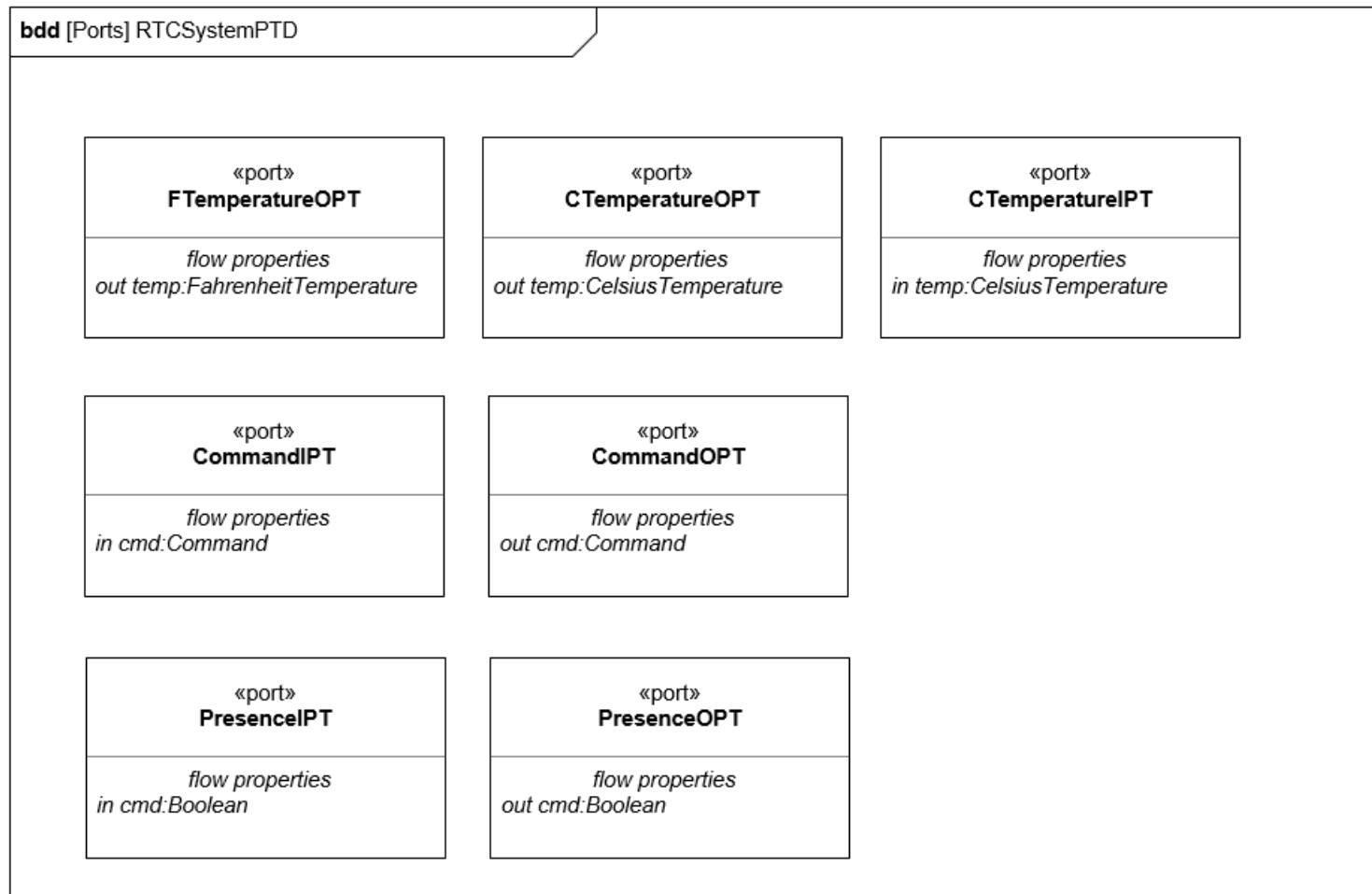
86

Port type definition

- We define the bdd of the RTC System port types (next slide)
- We define seven port types that use the value types defined in previous slides (*CelsiusTemperature*, *FahrenheitTemperature*), the *Command* enumeration and a Boolean type
 - three port types to provide temperature information
 - *FTemperatureOPT* is an out port type to provide temperature in Fahrenheit
 - *CTemperatureOPT* is an out port type to provide temperature in Celsius
 - *CTemperatureIPT* is an in port type that require temperature in Celsius
 - the *CommandIPT* and *CommandOPT* port types to respectively require and provide *On* and *Off* information
 - the *PresenceIPT* and *PresenceOPT* port types to respectively require and provide boolean data

Port types of the RTC architecture description

SysADL Notation



Component type definitions

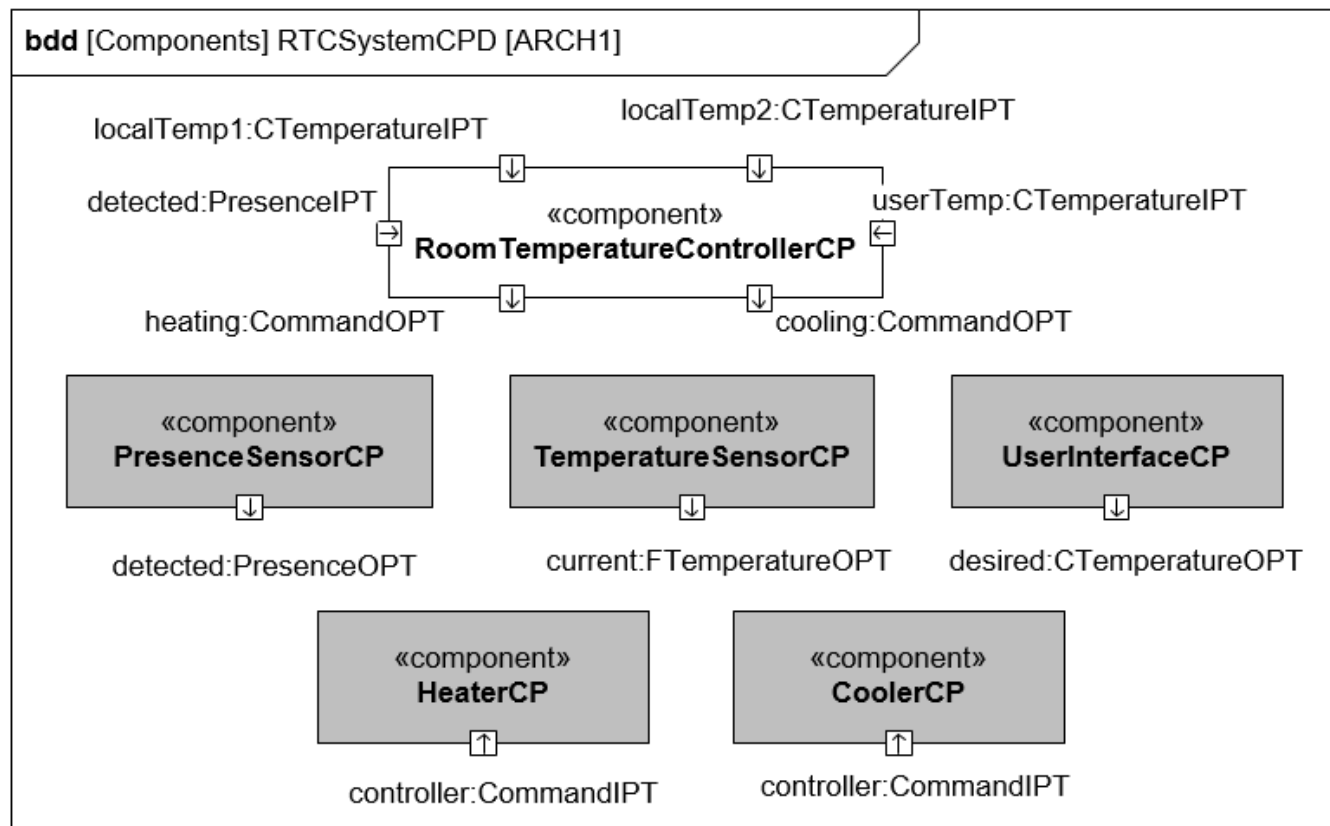
88

Component type in RTC System

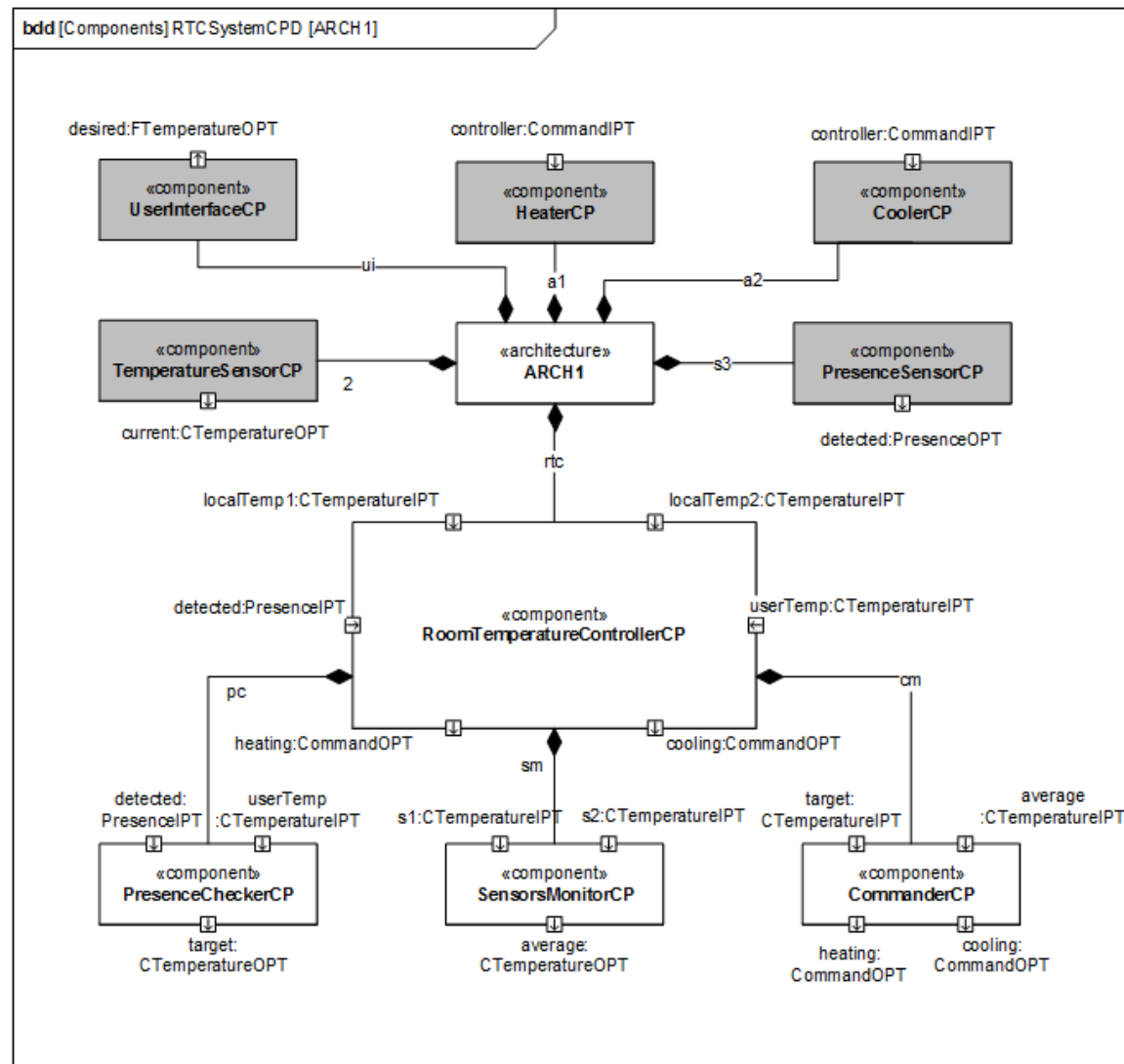
- We define the bdd of the RTC System component types and their ports
- We define six component types (see next slide)
 - *PresenceSensorCP*, *TemperatureSensorCP*, *HeaterCP*, *CoolerCP*, *UserInterfaceCP*, and *RoomTemperatureControllerCP*
- The *RoomTemperatureControllerCP* component type has 6 ports
 - all ports have name and a previously defined port type

Component types of the RTC architecture description

SysADL Notation



+ The component types for the configuration of RTC System



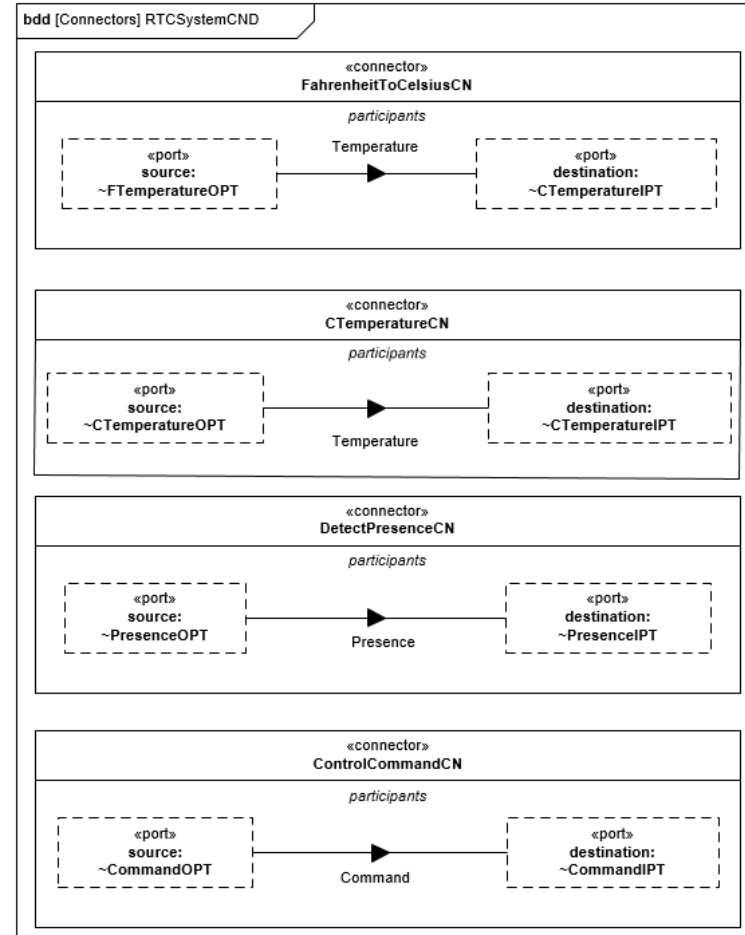
Connector types of the RTC architecture description

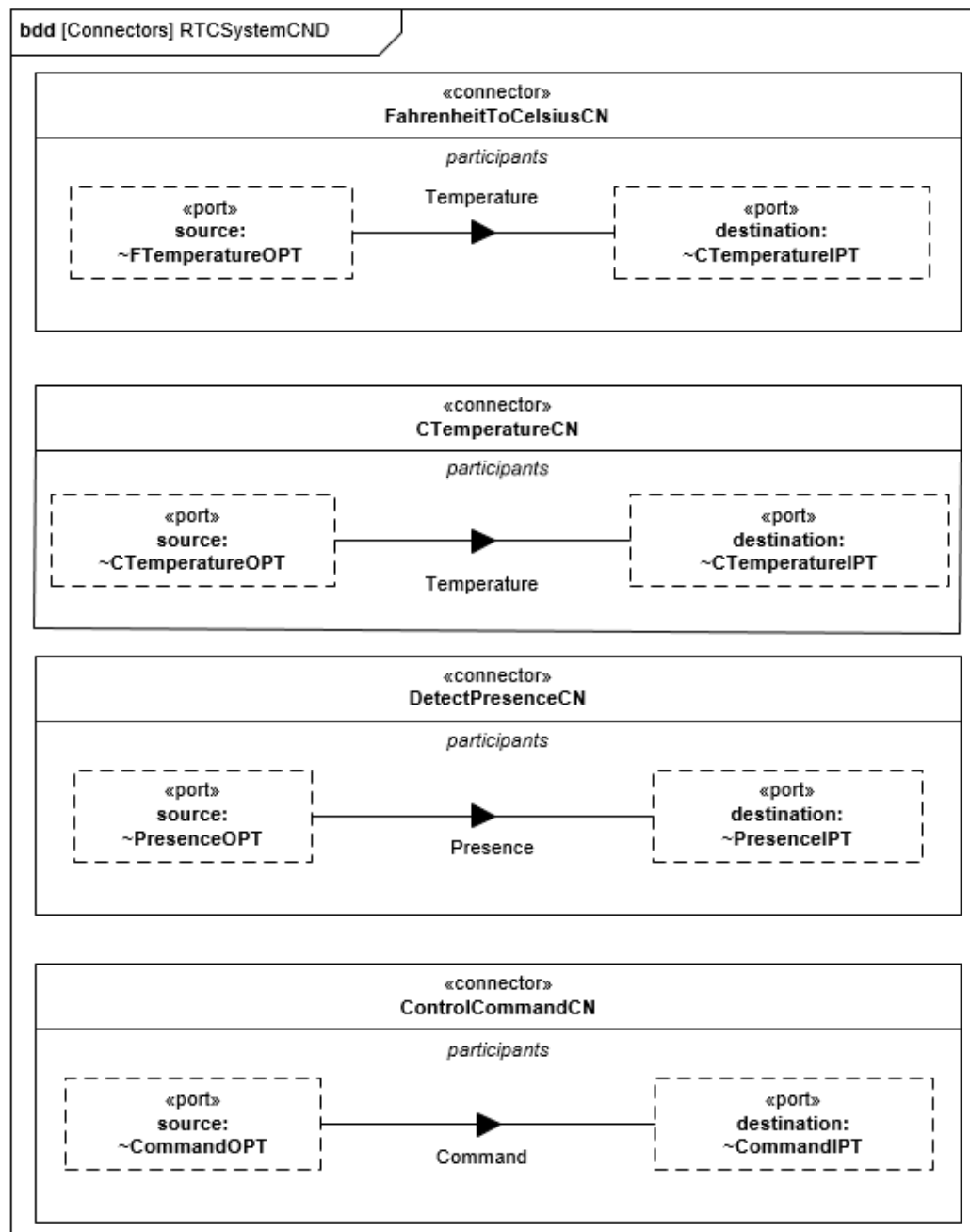
91

Connector types definitions

- We define the bdd of the RTC System connector types
- We define four connector types (see next slide)
 - *FahrenheitToCelsiusCN*, *DetectPresenceCN*, *CTemperatureCN*, and *ControlCommandCN*
 - all connector types in our example has two ports as participants
 - all ports and the type of information (value type or data type) that flows in the connection were previously defined
- The *FahrenheitToCelsiusCN* connector type has the responsibility of converting the data type from Fahrenheit to Celsius (details in the next chapter)

SysADL notation

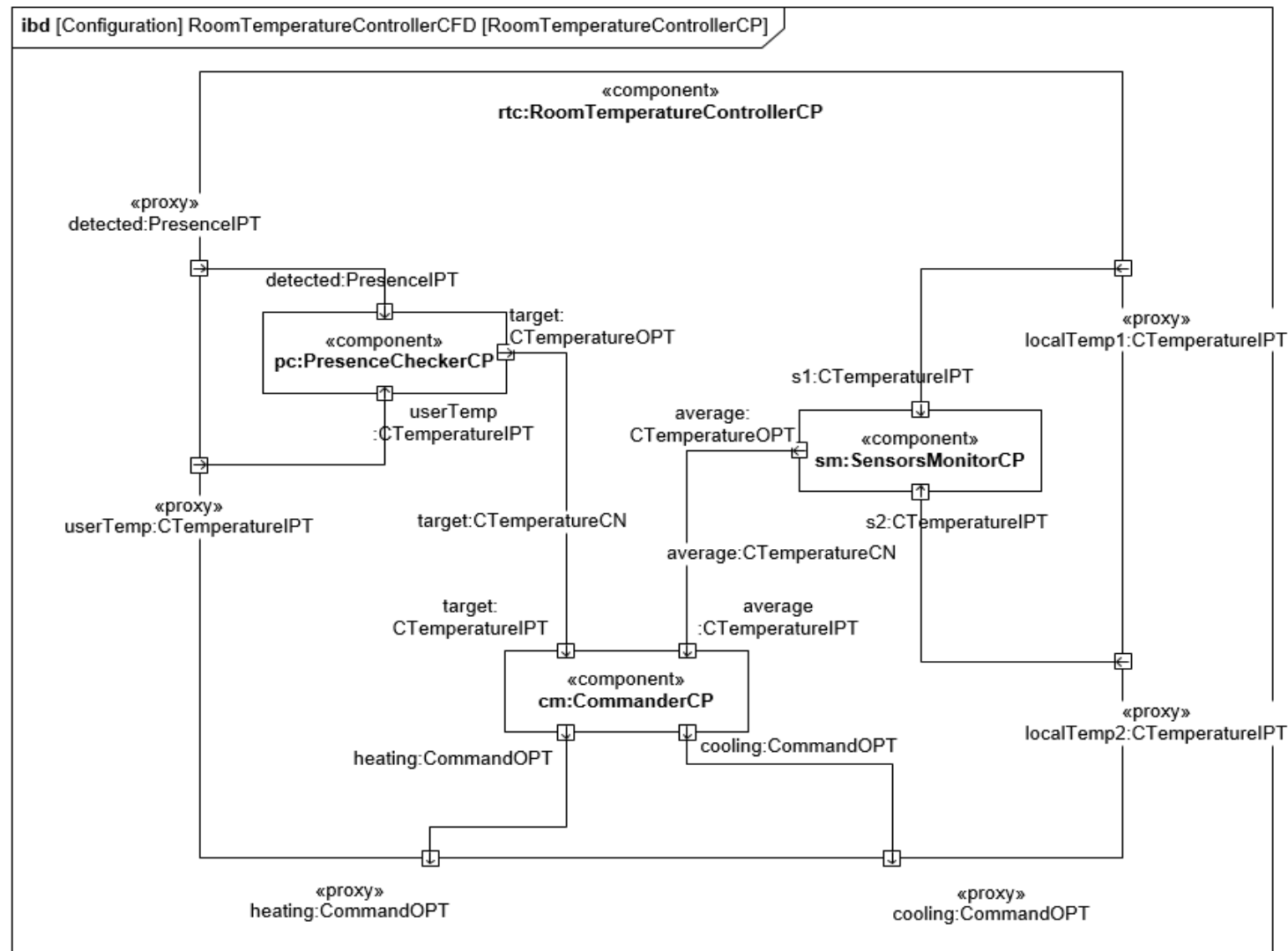




Configuration of the rtc composite component – description

- The *rtc* of *RoomTemperatureControllerCF* composite component type includes three components and two connectors:
 - *pc* – a *PresenceCheckerCP* component type;
 - *sm* – a *SensorsMonitorCP* component type;
 - *cm* – a *CommanderCP* component type;
 - *target* and *average* connectors – both of *CTemperatureCN* connector type.

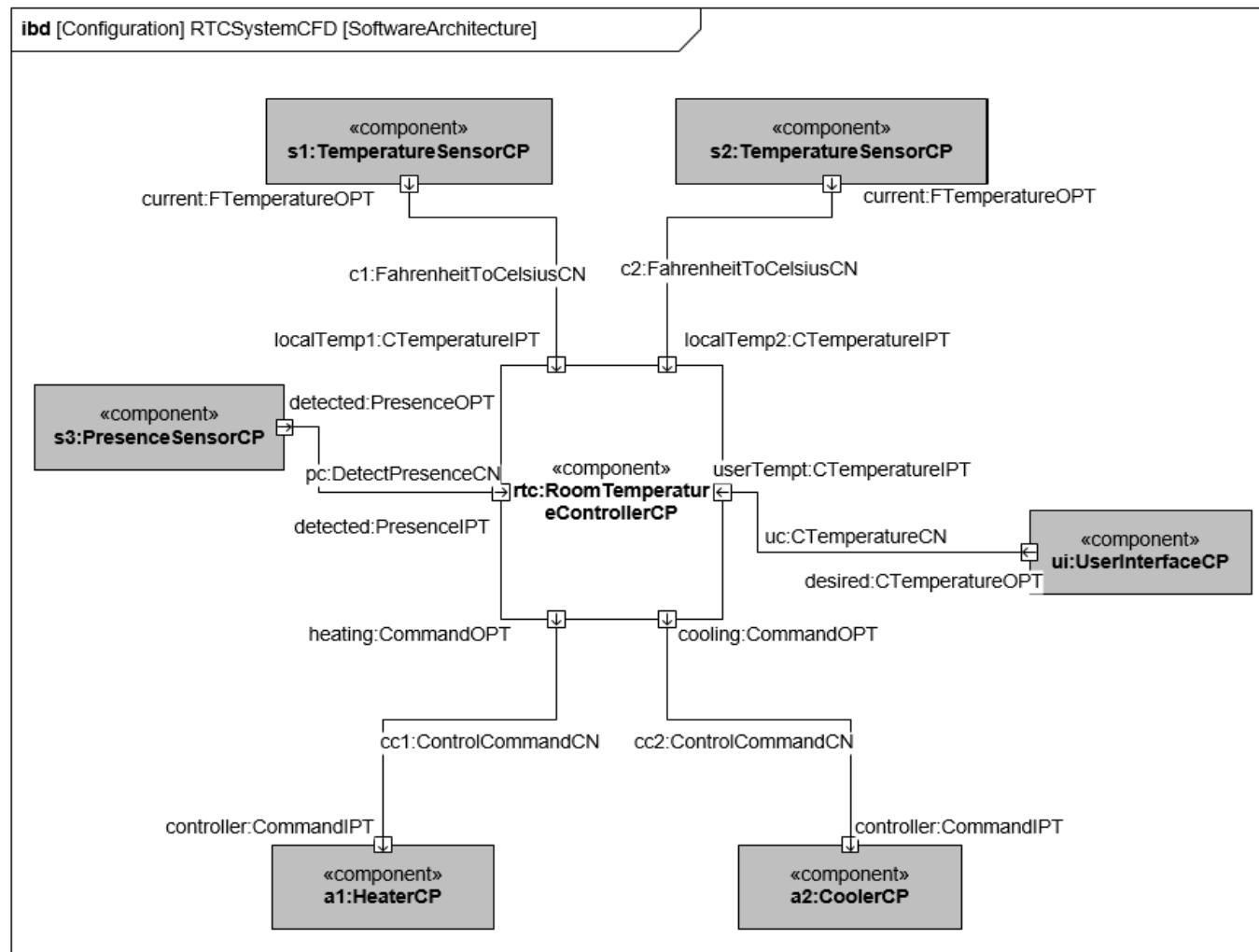
+ Configuration of the rtc composite component – illustration



Configuration of the RTC System Architecture – description

- The Software Architecture of the RTC System uses the elements defined previously:
 - *two sensors s1 and s2 of the TemperatureSensorCP boundary component type,*
 - *a s3 sensor of the PresenceSensorCP boundary component type,*
 - *a a1 actuator of the HeaterCP boundary component type,*
 - *a a2 actuator of the CoolerCP boundary component type,*
 - *a ui component of the UserInterfaceCP boundary component type, and*
 - *the rtc:RoomTemperatureControllerCP composite component*

+ Configuration of the RTC System Architecture – illustration



Summary

- In this chapter, you learned how to:
 - define and use the main structural constructs of a software architecture: components, connectors, and configurations;
 - define and use the auxiliary constructs applied to define the structural concepts: ports, value types, dimensions, and units;
 - organize these architectural definitions in the form of block definition diagrams and internal block diagrams specifying different structural views of the software architecture;
 - apply the SysADL notation to express the different constructs provided by the structural viewpoint.

For further reading

- Clements, P.; Bachmann, L.; Garlan, D.; Ivers, J.; Little, R.; Merson, P.; Nord, R. Documenting Software Architecture: Views and Beyond. SEI Series in Software Engineering. (2003)
- Hofmeister, C.; Nord, R.; Soni, D.; Applied Software Architecture, Addison-Wesley Professional. (1999).
- Taylor, R. N. ; Medvidovic, N.; Dashofy, E. M.; Software Architecture: Foundations, Theory, and Practice, 1st Edition. Wiley. (2010).