

# Software Architecture in Action

Flavio Oquendo, Jair C Leite, Thais Batista



# Chapter 2

## Viewpoints for Describing Software Architecture

# Learning outcomes of this chapter

- You will learn:
  - the definition of software architecture;
  - the concepts of viewpoints and views for describing a software architecture;
  - the architecture framework provided by SysADL in terms of concepts, viewpoints, and views

# The structure of the chapter

- The definition of Software Architecture
- Software Architecture Description
- Concepts for describing software architecture
- Architectural viewpoints and views
  - Architectural viewpoints
  - Architectural views
- Summary



# The Definition of Software Architecture

# Software Architecture

## Definition – 1/2

- The ISO/IEC Standard 42010 “Systems and Software Engineering — Architecture description” defines software architecture as
  - “*The fundamental properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution*”.

# Software Architecture

## Definition – 2/2

- As we can see, this definition highlights three matters in an architecture:
  - the **elements**;
  - the **relationships** between elements;
  - the **principles in the design and evolution** of these interrelated elements.
- This definition also highlights that the architecture comprises the **implied properties** emerging from these constituents.

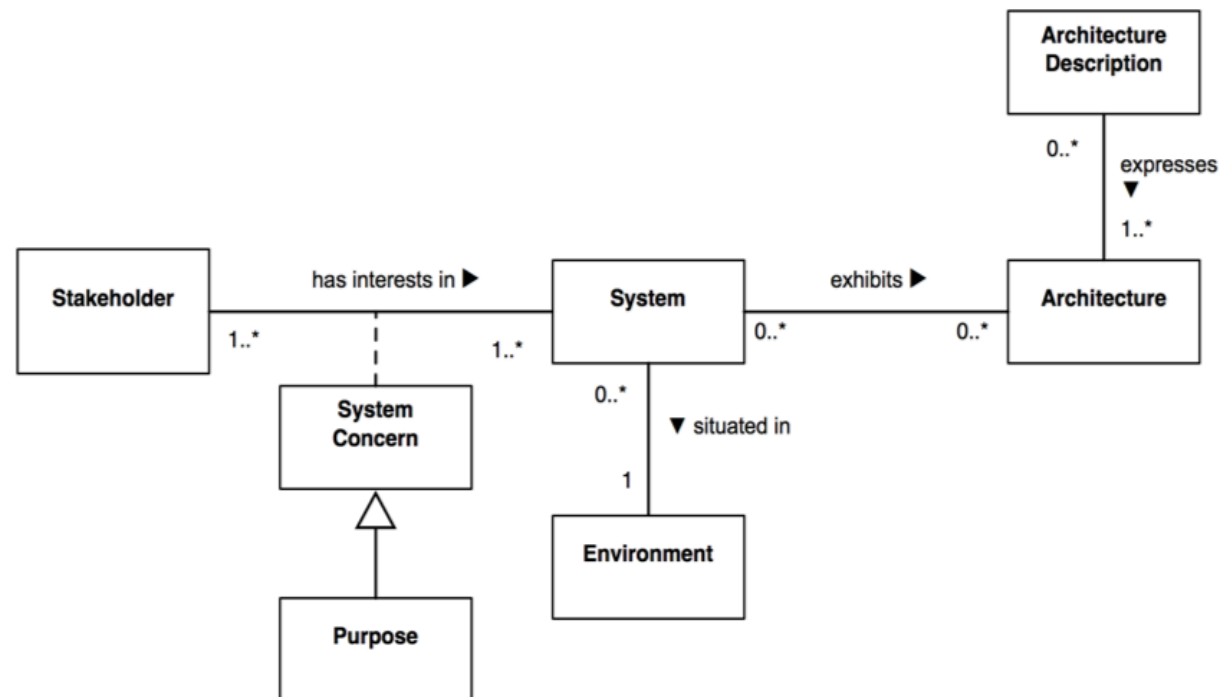


# Software Architecture Description



# Software Architecture Description

- According to the ISO/IEC/IEEE 42010 Standard, an architecture description is: “a work product used to express an architecture”.



# Software Architecture Description

## Concepts

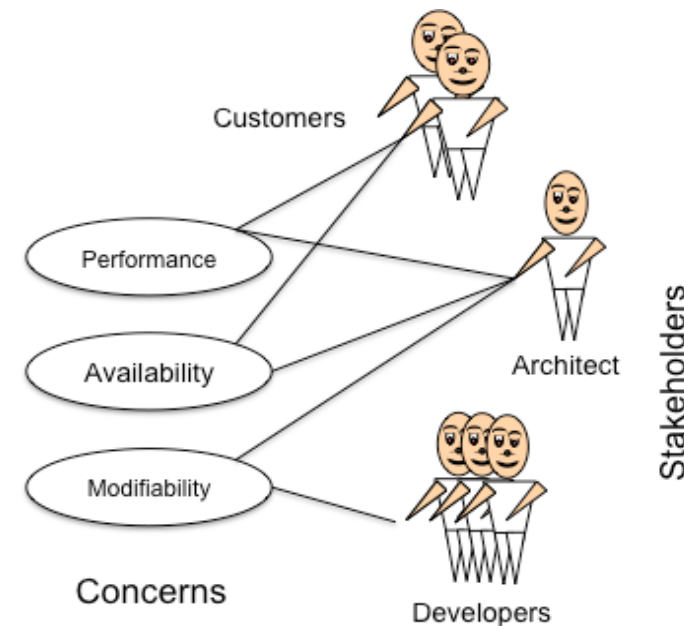
- The conceptual model of the context of an architecture description:
  - An architecture description expresses, **at least, one architecture**. It means that an architecture description may describe one or many architectures.
  - An architecture may be exhibited by none or several systems. It means that an architecture is a notion that **may be realized in different concrete systems**.
  - A **system** is situated in an **environment**. It means that the interaction with the environment needs be taken into account in the architecture description. Note that a system is more than the software part of it. Indeed, it is a **software-intensive system**.
  - The stakeholders comprise person, group, or organization with an interest in one or several systems. The interests are expressed as **concerns**. Each concern has a **purpose**.

# Example in the RTC System

11

## Stakeholders and Concerns in the RTC System

- Considering the RTC system, **stakeholders** are:
  - **Clients:** users of the system
  - **The architect:** in charge of developing a solution that satisfies client and developers concerns
  - **Developers:** concerned with modifiability, in order to facilitate maintenance and also to reduce the costs
- Associated **purpose** for:
  - **availability** is to enable the use of the system whenever need
  - **Performance** is the effectiveness and efficiency of the system



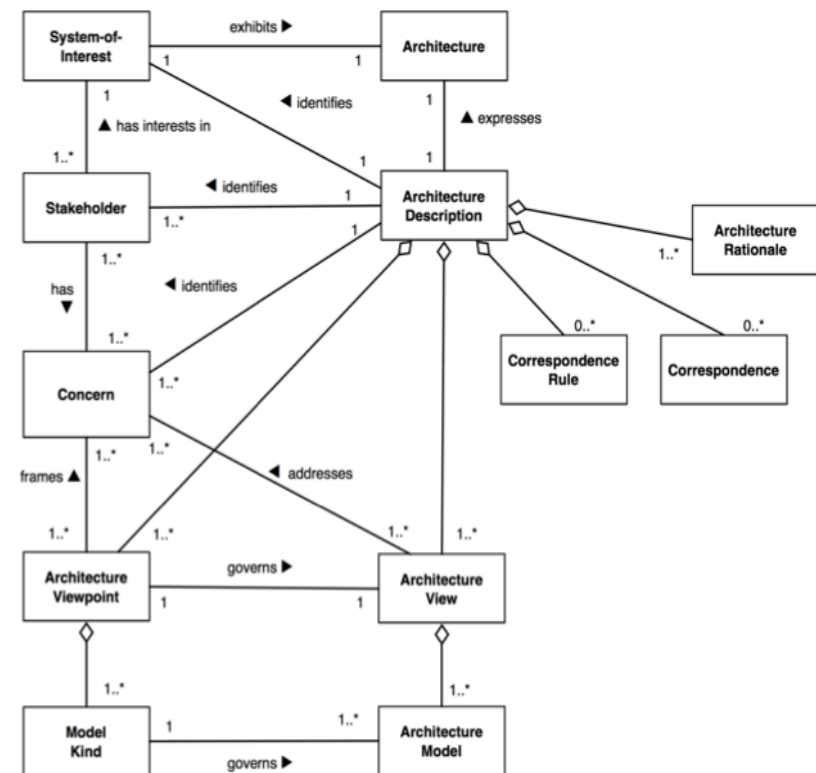
# Conceptual model of an architecture description (1/2)

ISO/IEC/IEEE 42010

12

## ■ An architecture description identifies

- a **system-of-interest**, for instance, a system that will be developed based on the described architecture;
- the system **stakeholders** and their concerns
- the **architecture viewpoints** used to represent the architecture

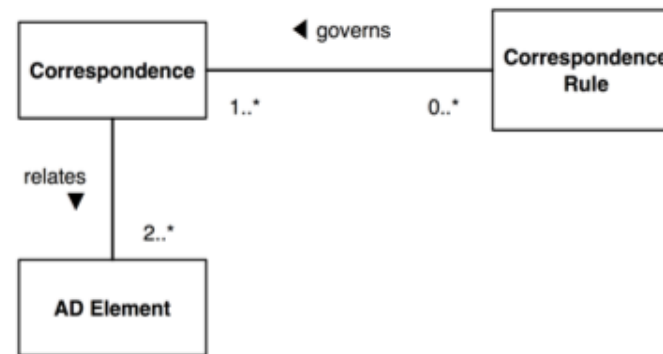


# Conceptual model of an architecture description (2/2)

ISO/IEC/IEEE 42010

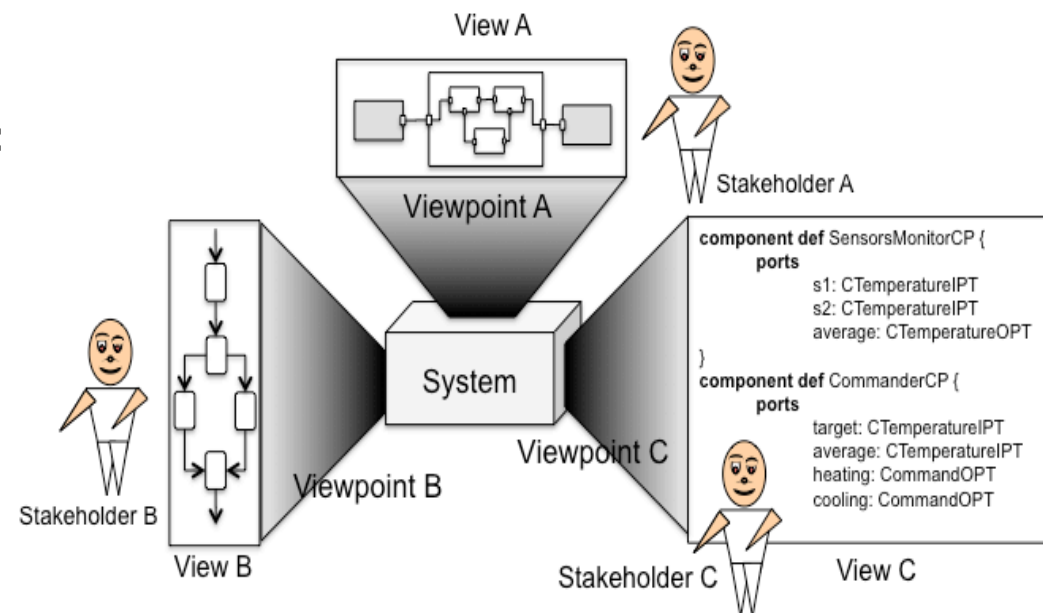
13

- An architecture description
  - **supports the correspondences between views** in terms of architectural elements, possibly defined by correspondence rules
  - **documents the architectural decisions** with their rationale



# + Viewpoints

- All concerns are mapped onto **viewpoints**
- Each **architectural viewpoint** governs one or more architecture **views**.
- **Architecture viewpoints** defines the constructs, including rules and conventions, for the definition of the views
- An **architecture view** comprises one or more architecture



3 viewpoints,  
each one with a view.

# Architect role

- An architect has the role of
  - **describing the architecture** of the system-of-interest, considering the environment, the stakeholders and their concerns.
  - describing the architecture using different **architecture views** according to the viewpoints supported by an *architecture framework*.

# Architecture Framework

## Definition

- According to the ISO/IEC Standard 42010, an architecture framework
  - provides the “*conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders*”.



# Architecture Framework

## SysADL

- SysADL is an **architecture framework** for software engineers and developers, providing a set of viewpoints to the stakeholders.
- For each viewpoint, the SysADL framework **defines a language with conventions and principles** about the definition and use of the architectural elements of the viewpoint.



# Concepts for describing software architecture

# Architecture Description

## Main building blocks

- The ISO/IEC/IEEE 42010 Standard specifies a conceptual model of architecture description,
  - but **it does not define the main building blocks and primary elements** of an architecture description
- For this purpose, we adopt the well-known abstractions of **components, connectors, and configuration**.
  - these concepts are **independent** of any specific notation, concrete syntax, or architecture language

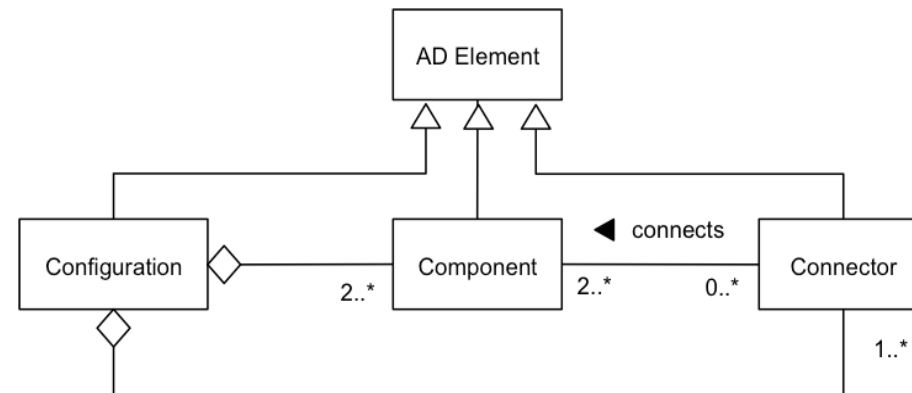
# Architecture Description

## Main building blocks

### Concepts

- An architecture description element (**AD element**) may be
  - a component
  - a connector
  - a configuration
- A **configuration** may contain one or more components, and none or several connectors
- A **connector** connects two or more components

### Conceptual model



# Architecture Description Element

## Components

- **Components** are architectural elements that provide functionalities in a system
  - The central element
  - The loci of computation and state
- The figure below illustrates two components
  - Temperature sensor
  - Presence sensor



# Architecture Description Element

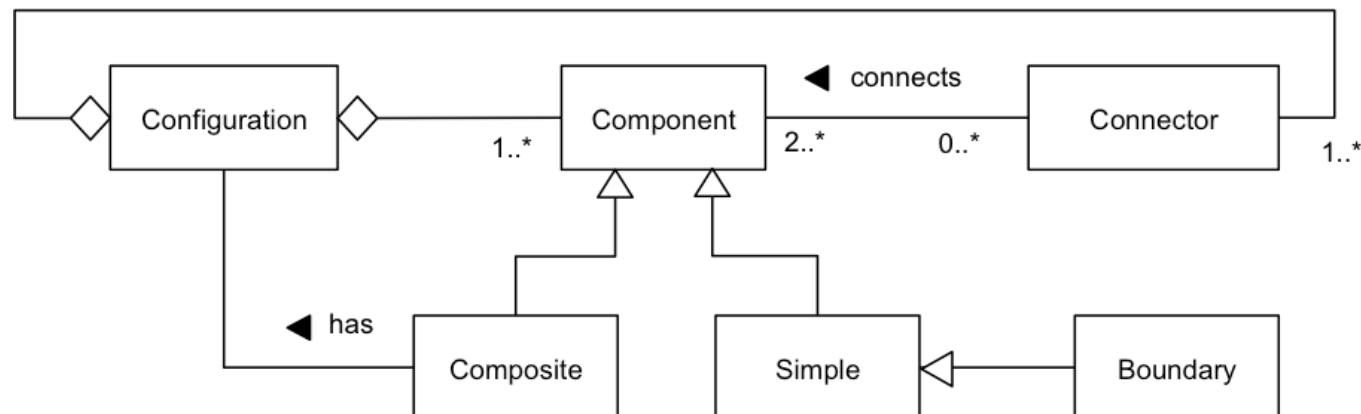
22

## Simple and composite components

### Concepts

- A component can be
  - a **simple** element providing a simple functionality
  - a **composite** element representing a system as a whole, which itself is composed of others components
- A **simple component** performs sequential computation using data available in **ports**
- A **composite component** performs concurrent computation by being composed of components in their internal structure

### Conceptual model



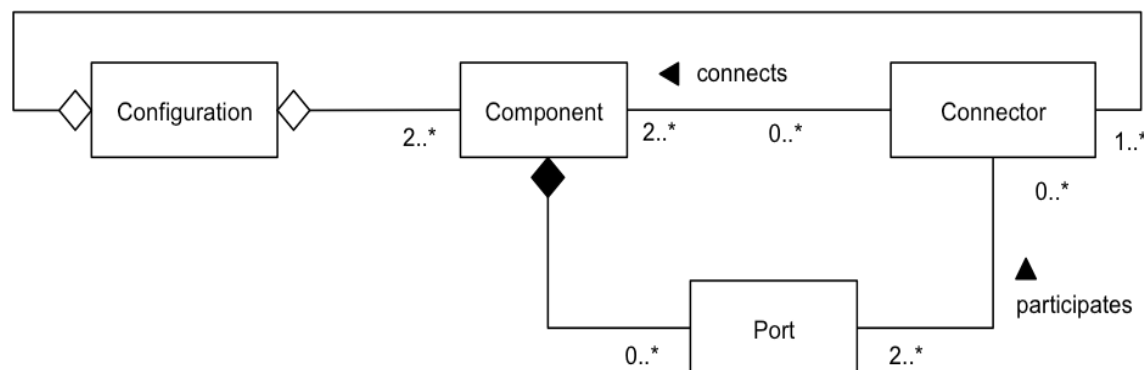
# Architecture Description Element

23

## Ports

- A component has clearly defined **ports** that are interaction points between it and the external world, i.e. its environment
  - A component has none or several ports
  - Ports specify the data that a component **provides** or **requires** from other components in the architecture
  - The explicit specification of the data that a component provides and requires is essential to support the **component's composability**
- **Ports** belong to components and participate to connectors (ports do not belong to connectors)

### Conceptual model



# Architecture Description Element

## internal or boundary components

- Components can be
  - **Internally** located
  - **Boundary** between the system and the environment
- **Boundary components** are placed at the interface with the environment, encapsulating the mechanisms the system uses to interact with the environment.
- **Internal components** are located inside a composite component including the architecture itself and therefore are not visible outside of it.
- The RTC system has different boundary components such as
  - a sensor that physically measures the temperature and provides its value in an *out* port.



# Architecture Description Element

## Connector

- A **connector** is an element responsible for **mediating the interaction** among components, establishing rules that govern those interactions.
- It provides the **glue mechanisms** to binding components together
- Each side of the connector specifies the kind of component's port that can be bound to it



# Architecture Description Element

## Connector

- Connectors define
  - which **ports** can be connected
  - how the **interactions** between connected components take place
- Connectors are the **locus of communication**
- Connectors **connects at least two ports**

# Architecture Description Element

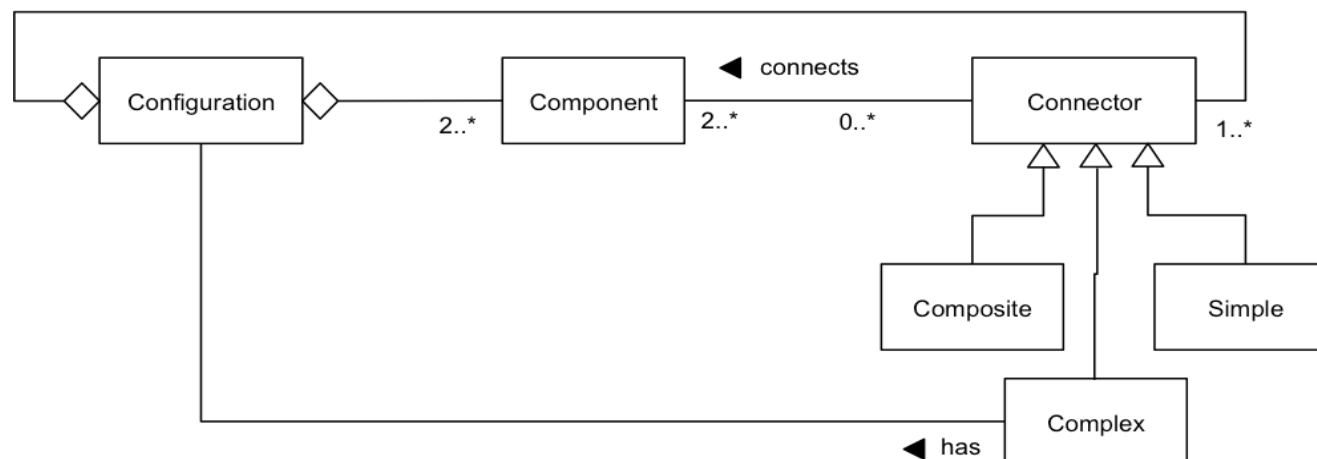
## Connector

27

### Concepts

- A connector can be
  - A **simple element** connecting two or more ports
  - A **composite element** which itself is composed of others connectors.
  - A **complex element** that has a configuration
- In the case of the RTC system, a connector could be used to connect the temperature sensor with the controller component

### Conceptual Model



# Architecture Description Element Configuration

- **Configurations** describe the **topology** for identifying
  - which components are part of a software architecture, and
  - how they are connected together through connectors.
- It defines a **connected graph** of components and connectors

# Architecture Description Element

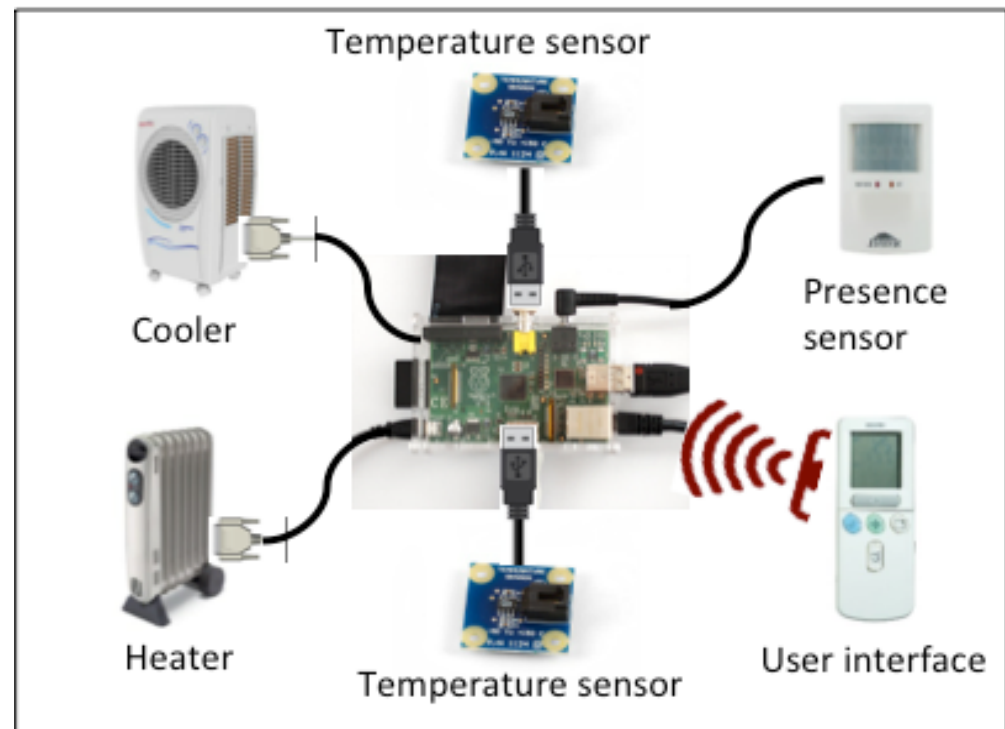
29

## Configuration example

### Example

- In the RTC system, the configuration could define that the controller is connected in a star topology configuration with
  - two temperature sensors,
  - one presence sensor,
  - and two actuators – the cooler and the heater.
  - the use cable connectors to send and receive data to the controller.
  - the user interface is a remote control that sends data via an infrared connector

### Illustration



# Architecture Description Element

- These three concepts raise different needs in terms of software architecture description:
  - how to describe the ports of components?
  - how to describe the ports of connectors?
  - how to describe the configuration of components and connectors?
  - how to describe the behavior of components?
  - how to describe the behavior of connectors?
  - how to describe the behavior of the configuration of components and connectors? And also:
  - how to validate the designed architecture?
  - how to verify the implied properties of the designed architecture?

**Architecture description languages (ADLs) addresses these issues.**

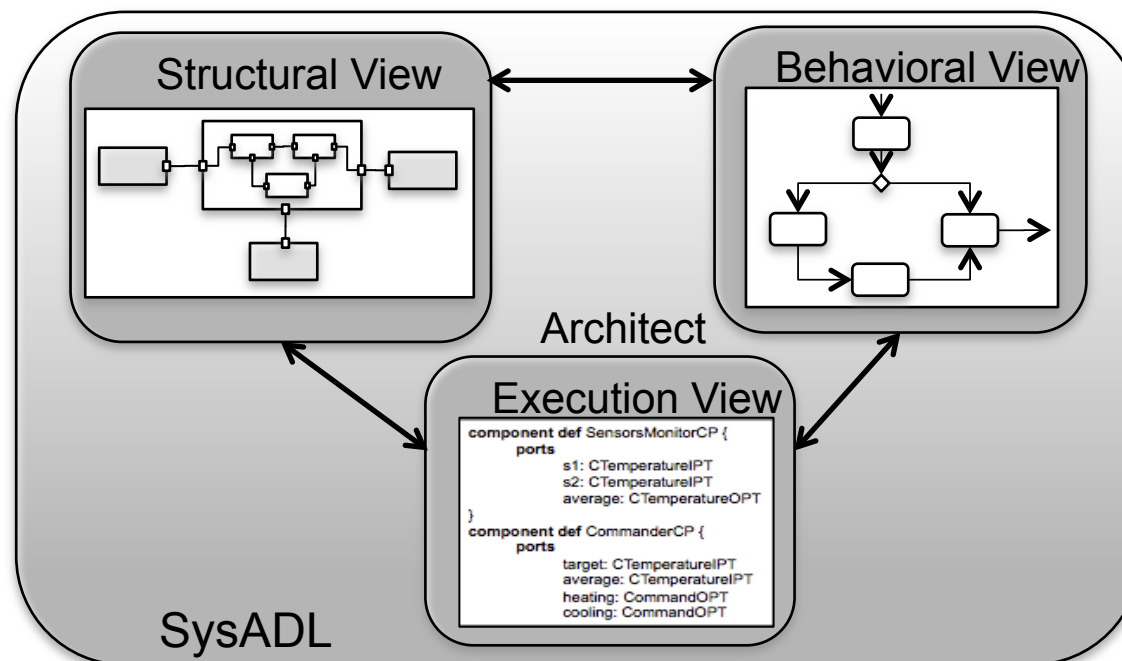


# Architectural Viewpoints and Views

# Architecture Description

## Viewpoints and Views

- An architecture is described from different **viewpoints** in terms of **views** and represented by **model diagrams**.







# Architecture Viewpoints

# Architecture Viewpoints

## Concept

- Viewpoints realize the stakeholders concerns by means of **architecture views**.
- A **viewpoint** defines the kinds of model that govern the diagrams used to represent its corresponding views. Such models are what a stakeholder “sees” when looking at the system from a specific viewpoint.
- The ISO/IEC/IEEE 42010 Standard also emphasizes that **multiple views** are essential to cover all the stakeholders’ concerns, and to detail the architecture from different perspectives.

# Architecture Viewpoints

## Analogy between software architecture and civil architecture

- The well-known structural, electrical, and hydraulic plans are some **examples of civil architecture views** from the structural, electrical, and hydraulic viewpoints.
- Each one **provides a specific view of a building.**
- The **plans are used by the stakeholders** – engineer, architect, brick worker – to reason about and govern the building process.
  - They help the stakeholders to understand the building concerns.
- Similarly, software architecture views show **different perspectives of the software-intensive system.**
  - They communicate the architecture to the different stakeholders.

# Architectural Viewpoints and Stakeholders

36

## Example – 1/2

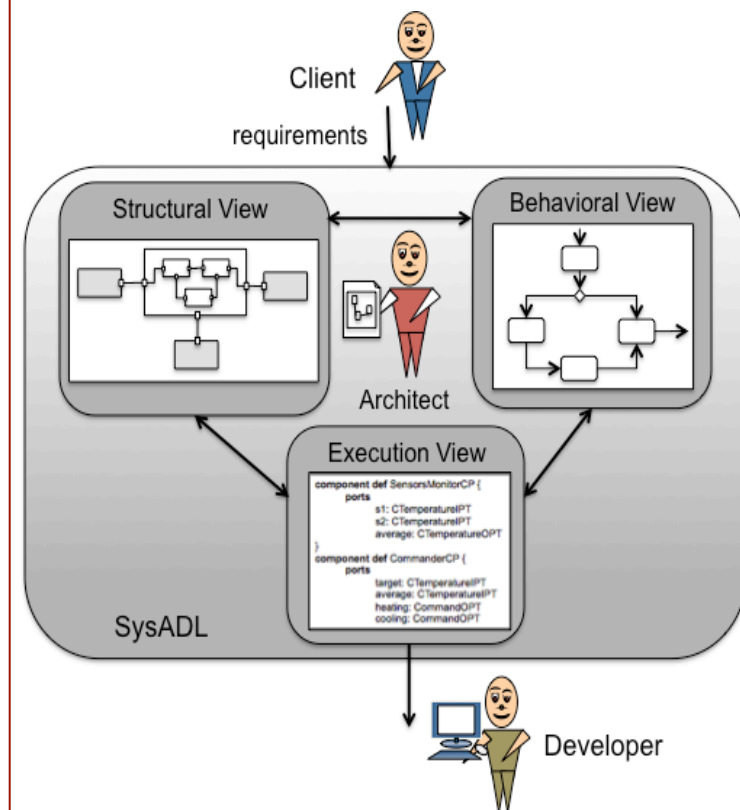
- The main stakeholders interested in software architecture description are clients, architects, and developers

- **Client**

- specifies the requirements of the system to be realized by the architecture
- is interested in the executable architecture satisfying the requirements

- **The architect**

- is concerned with the specification of the architecture, reasoning about it to verify properties and detailing its structure, behavior, and execution

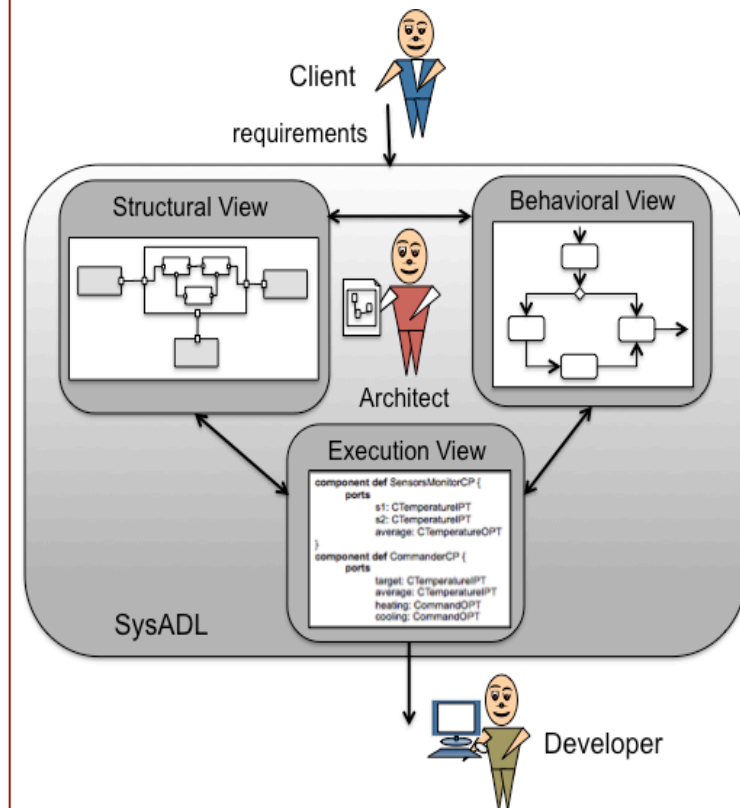


# Architectural Viewpoints and Stakeholders

37

## Example – 2/2

- (Continued)
  - **The developer**
    - is interested in the executable architecture for implementing the system.
  - Structure and behavior are **declarative specifications** that can be mapped to a corresponding executable model, characterizing the executable viewpoint



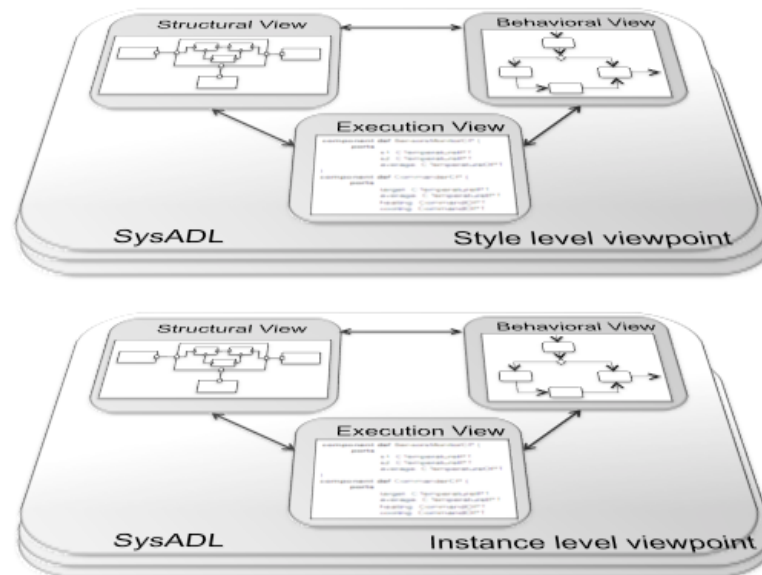
# Architectural Views

## In SysADL

- SysADL defines three architectural viewpoints that are essential to communicate the software architecture to the involved stakeholders:
  - the structural viewpoint;
  - the behavioral viewpoint;
  - the executable viewpoint.

# + Architecture Views and Viewpoint

- Each viewpoint provides the **constructs for describing different views** of the architecture from these viewpoints
- Two viewpoints (**style level viewpoint** and **instance level viewpoint**) and an architecture description with three views (one for each of the three viewpoints)



# Architecture Views and Viewpoint

## Structural View – 1/2

- The **structural view** describes the main building blocks of the architecture from a conceptual point of view:
  - components
  - connectors, and
  - configurations.
- It makes use of **two kinds of diagrams**:
  - the **block definition diagram** (*bdd*), and
  - the **internal block diagram** (*ibd*).



# Architecture Views and Viewpoint

## Structural View – 2/2

- The *bdd* is used
  - to define the components and connectors of the architecture.
- The *ibd* goes a step further
  - showing how components and connectors bind each other defining the configuration of the architecture or of the internal structure of composite components and connectors.

# Architecture Views and Viewpoint

## Behavioral View

- The **behavioral view** is concerned with the behavior of the architecture, specifying
  - sequential and concurrent activities by making use of the **activity diagram (*act*)**.
- The **behavioral description** also includes
  - the **specification of protocols** supporting the interaction between components through connectors, and
  - the **specification of actions** using the **parametric diagrams (*par*)**.

# Architecture Views and Viewpoint

## Executable View

- The **executable view** describes
  - the execution details of the architecture.
- It represents the **code view**.
- This viewpoint provides a superset of the **OMG Action Language for Foundational UML (ALF)** as part of SysADL.
- The use of ALF allows for a **direct mapping** onto the programming language level.

# Architecture Views and Viewpoint

## Style level and instance level (1/3)

- In addition to views, we have two abstraction levels:
  - style level, and
  - instance level.
- The *style level* is
  - where the **architecture style** is specified, defining the types of components and connectors that compose the architecture, as well as the set of constraints on how they are combined.
  - It is used to derive **instances of the architecture**, following the types and constraints that are defined.

# Architecture Views and Viewpoint

## Style level and instance level (2/3)

- For instance, the **client-server style** defines
  - **server** and **client** as component types,
  - a **server-client connector type** to connect servers and clients, and
  - **constraints** on systems composition specifying the kinds of allowed compositions.
    - For example, clients must not communicate among them, clients communicate only with servers, servers may not initiate the communication with a client, and may allow no more than 10 clients to be simultaneously connected to them.

# Architecture Views and Viewpoint

## Style level and instance level (3/3)

- The *instance level* is
  - where the **instances of the style** are defined.
  - For each style, **several instances** can be defined.
  - An instance is **specified by instantiating** the elements that compose the style, and by satisfying the constraints.
  - For the client-server style
    - a possible simple instantiation can be a **Web server connected to different clients** using one connector for linking each client to the server.
    - another instantiation can be a **banking server with a dedicated connector** for each client.

# Summary

- In this chapter, you learned:
  - how to define a software architecture;
  - what are the underlying concepts needed for describing a software architecture: viewpoints and views;
  - what is the architectural framework provided by SysADL in terms of viewpoints and abstraction levels.

# For Further Reading

- Bass, L., Clements, P., Kazman, R.: Software Architectures in Practice, 2nd edn. Addison Wesley, Reading (2003)
- 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)
- Rozanski, N., Woods, E.: Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. Addison-Wesley (2012)